(4)

AUGUST 1989 LIDS-R-1901

Research Supported By:

Office of Naval Research
Grant N00014-85-K-0782

REQUIREMENTS SPECIFICATION WITH PETRI NETS USING THE CUBE TOOL METHODOLOGY

Didier Marie-Joseph Peidu

Laboratory for Information and Decision Systems
MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAMBRIDGE, MASSACHUSETTS 02139

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) LIDS-R-1901	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Lab. for Inf. and Dec. Systems 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) Mass. Institute of Technology Room 35-410, LIDS Cambridge, MA 02139	7b. ADDRESS (City, State, and ZIP Code) 800 N. Quincy Street Arlington, VA 22217-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-85-K-0782
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. NR564-001 / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
REQUIREMENTS SPECIFICATION WITH PETRI NETS USING THE CUBE TOOL METHODOLOGY

12. PERSONAL AUTHOR(S)
DIDIER MARIE-JOSEPH PERDU

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
technical	FROM _____ TO _____	August 1989	77

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The distributed nature of command and control requires the consideration of both processes and communications in the formulation of requirements. Cube Tool is a methodology used to derive the processing and communication needs for each system function. An approach is introduced for extending the applicability of Cube Tool to the determination of requirements for C3I systems. First, using Cube Tool for each function, a Petri Net is derived that models all processes and communications for the correct execution of the function. Then, for a given scenario, these nets are interconnected and the steps of the methodology are applied again to derive the Petri Net that represents the mission-dependent requirements for the system. For different modes of operation are considered, different Petri Net are obtained which can then be folded together to obtain a Colored Petri Net representation of the processes and communication needs.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED-UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL CDR T. WELLS	22b. TELEPHONE (Include Area Code) (202) 696-4713
	22c. OFFICE SYMBOL Code 1224

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted; all other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

AUGUST 1989 LIDS-R-1901

REQUIREMENTS SPECIFICATION WITH PETRI NETS USING THE CUBE TOOL METHODOLOGY

by

DIDIER MARIE-JOSEPH PERDU

Ingénieur de l'Ecole Supérieure d'Electricité (July 1986)

Master of Science in Technology and Policy
from the Massuchusetts Institute of Technology (February 1988)

The research was conducted at the MIT Laboratory for Information and Decision Systems with support provided in part by THOMSON-CSF, CIMSA SINTRA Division, Colombes, France and in part by the Basic research Group of the Joint Directors of Laboratories, through the Office of Naval Research, under Contract No. N00014-85-K-0782.

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge MA 02139

REQUIREMENTS SPECIFICATION WITH PETRI NETS USING THE CUBE TOOL METHODOLOGY

by

DIDIER M. PERDU

ABSTRACT

The distributed nature of command and control requires the consideration of both processes and communications in the formulation of requirements. Cube Tool is a methodology used to derive the processing and communication needs for each system function. An approach is introduced for extending the applicability of Cube Tool to the determination of requirements for C3I systems. First, using Cube Tool for each function, a Petri Net is derived that models all processes and communications for the correct execution of the function. Then, for a given scenario, these nets are interconnected and the steps of the methodology are applied again to derive the Petri Net that represents the mission-dependent requirements for the system. For different modes of operation are considered, different Petri Nets are obtained which can then be folded together to obtain a Colored Petri Net representation of the processes and communication needs.

Research Supervisor: Dr. Alexander H. Levis
Title: Senior Research Scientist

ACKNOWLEDGEMENTS

I wish to express my gratitude to:

Alexander H. Levis, for his remarkable supervision in the realization of this report. By always demanding from himself what he demands from his students, he taught me much more than Distributed Intelligent Systems: a sense of hard work, commitment, excellence. He showed me what research is about and working for him these last three years has really been a rewarding experience.

Messrs. Christian Tournes, Alain Bourrez and Thubert from THOMSON-CSF, CIMSA-SINTRA Division for their help.

Cindy, François, Jacques, Jinane, Louise, Sabina, Vicky and Stamos for their comradeship and advice.

My parents and grandparents for having made this experience possible.

And last but not least, Joanne for her love and devoted support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 PROBLEM DEFINITION

The information and data handled by each part of a C3I (Command, Control, Communication and Intelligence) system are complex and numerous and it is difficult for the commander to get an accurate view of what happens on the battlefield. Information has to be fused in order to avoid the accumulation of useless or redundant data. Another problem is that each part of the system has its own particular interests and concerns in his battlefield area which are not always the main concern or the interest of the system in a whole. Therefore, a need exists to coordinate all these parts to make the system effective and this coordination has to be taken into account at the design stage with the formulation of the requirements

The determination of the functional requirements of a system is usually done by representing the relationships among the different processes which have to take place for the execution of a mission. C3I systems are, among other characteristics, geographically dispersed and this dispersion of the resources, processes and communication is a critical aspect. Therefore, requirements must include not only the processes, but also the communications among the different parts of the system.

The Cube Tool has been developed as a methodology for deriving the processing and communication needs for each system function. It is used to define the tasks to be performed, the resources needed to perform these tasks, and the time when these tasks are performed. Focusing on both processing and communication, this methodology can define clearly what the requirements are for each system function.

The aim of this report is to present an extension of Cube Tool for representing the requirements for a given scenario. It presents how to generate a Petri Net representation of the responsibilities for each function and then how to derive the Petri Net of the requirements for a scenario, when the functions are linked together to make possible the fulfillment of as specific mission. For different modes of operation, a Colored Petri Net representation of the requirements can be deduced by folding the Petri Net representation of requirements for each mode of operation.

1.2 THE REPORT IN OUTLINE

Chapter 2 describes briefly the theory of Ordinary Petri Nets and Colored Petri Nets, focusing only on the aspects which are used to represent the requirements of a system. Then, Chapter 3 introduces the Cube Tool and shows how the processing and communication needs are derived for each system function. In Chapter 4, a methodology for going from the Cube Tool formalism to a Petri Net representation of the requirements is presented. It is then extended to generate the Colored Petri Net representation of the requirements for several modes of operation. This methodology is applied to a realistic example in Chapter 5, where a system for Air Interdiction Mission Planning is specified using Cube Tool. The Petri Nets of the system functions are derived and merged to generate the Petri Net of the requirements of the system for a given scenario. Finally, conclusions are drawn in Chapter 6 and directions for future research are presented.

CHAPTER 2

PETRI NETS AND COLORED PETRI NETS

2.0 INTRODUCTION

Petri Nets (Peterson, 1980; Reisig, 1985) are used for the modeling and the analysis of concurrent and asynchronous processes. Their fields of application range from the modeling of manufacturing processes to the representation of the flow-charts of complex computer software. They have been successfully used for the modeling of decisionmaking organizations (Remy et al., 1988) since they provide an explicit representation of the interactions among decisionmakers.

Petri Nets have been introduced in the modeling of Distributed Systems because they give a graph-theoretic representation of the communication and control patterns, and a mathematical framework for analysis and validation. Petri Net modeling is appealing for the following reasons: Petri Nets provide an integrated methodology, with well developed theoretical and analytical foundations, for modeling physical systems together with complex cognitive decision processes, and they capture the precedence relations and structural interactions of concurrent and asynchronous events. Deadlocks and conflicts can be easily identified on a Petri Net since the graphical nature of Petri Nets helps to visualize easily the complexity of the system. Thus, they are appealing to the layman as well as to the analyst. Various extensions of the basic theory allow for quantitative analysis of resource utilization, throughput rate, effect of failures, and real time implementation.

First, the theory of Ordinary Petri Nets will be presented, followed by a discussion on its limitations and, finally, Colored Petri Nets (Jensen, 1986) will be introduced.

2.1 ORDINARY PETRI NETS

2.1.1 Definitions

Definition 2.1

A Petri Net - denoted by PN - is a bipartite directed graph represented by a quadruple PN = (P, T, I, O) where :

- P = {p1, ..., pn} is a finite set of n places. A place is depicted by a circle node and models a resource, a buffer, or a condition.
- T = {t1, ..., tm} is a finite set of m transitions. A transition is represented by a bar node and stands for a process, an event, or an algorithm.
- I is a mapping PxT -> {0,1} corresponding to the set of directed arcs - called connectors - from places to transitions. I(pi, tj) = 1 means that there exists a connector from the place pi to the transition tj which indicates that the process tj requires the availability of the resource pi, the fulfillment of the condition pi, or the availability of information in the buffer pi, in order to occur.
- O is a mapping TxP -> {0,1} corresponding to the set of connectors from transitions to places. O(tj, pi) = 1 means that there exists a connector from the transition tj to the place pi which indicates that when the process tj is finished, it either enables the condition pi, makes the resource pi available, or sends an item of information to the buffer pi.

An example of a Petri Net, PN1, is shown in Figure 2.1.

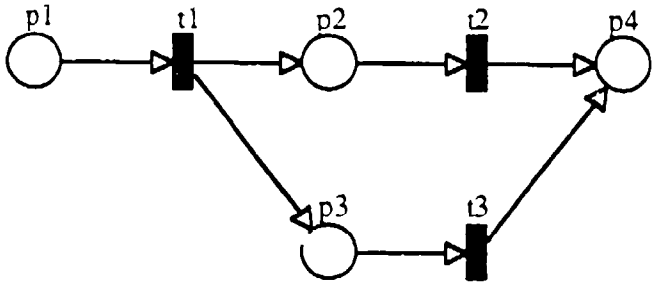


Figure 2.1 Petri Net PN1.

In this example, we have :

P = {p1, p2, p3, p4},

T = {t1, t2, t3},

I(p1, t1) = 1	I(p2, t1) = 0	I(p3, t1) = 0	I(p4, t1) = 0
I(p1, t2) = 0	I(p2, t2) = 1	I(p3, t2) = 0	I(p4, t2) = 0
I(p1, t3) = 0	I(p2, t3) = 0	I(p3, t3) = 1	I(p4, t3) = 0

O(t1, p1) = 0	O(t2, p1) = 0	O(t3, p1) = 0
O(t1, p2) = 1	O(t2, p2) = 0	O(t3, p2) = 0
O(t1, p3) = 1	O(t2, p3) = 0	O(t3, p3) = 0
O(t1, p4) = 0	O(t2, p4) = 1	O(t3, p4) = 1

Definition 2.2

A Petri Net is *pure* if and only if it has no self loops, i.e., no place that can be both an input and an output of the same transition.

The net of Fig. 2.1 is pure and all Petri Nets that are considered in this report are pure.

Definition 2.3

A *path* is a set of k nodes and k - 1 connectors, for some integer k, such that the i-th connector either connects the i-th node to the i+1-th node or the (i + 1)-th node to the i-th node. The path is *directed* if the i-th connector connects the i-th node to the (i + 1)-th node for all i = 1,..k..

For example, in Figure 2.1:

p1 - t1 - p2 - t2 - p4 is a directed path,

p4 - t3 - p3 is not a directed path.

If a Petri Net has sources and sinks, then any path from a source to the sink is called an *information flow path*. If an information flow path is a set of k nodes such that the k nodes are distinct, then the information flow path is said to be *simple*.

In the example shown on Figure 2.1, the source of the net is place p1, the sink is place p4 and the simple information flow paths are:

Simple path 1: p1 - t1 - p2 - t2 - p4

Simple path 2: p1 - t1 - p3 - t3 - p4

Definition 2.4

A Petri Net is *connected* if and only if there exists a path - not necessarily directed - from any node to any other node.

Fig. 2.1 depicts a connected net. Intuitively, this definition formalizes the idea that a Petri Net models a whole system. There are no partitions of the set of nodes into disjoint subsets, such that the nodes in one subset are not connected to the other subsets.

Definition 2.5

A Petri Net is *strongly connected* if and only if there exists a directed path from any node to any other node.

The net of Fig. 2.1 is not strongly connected as exemplified by the lack of directed path from p2 to p3. Nets with sources and sinks are not strongly connected.

Definition 2.6

The *slices* of a Petri Net (Hillion and Levis, 1987) are the sets of places or transition which represent concurrent activity in the process modeled by this Petri Net.

The net of Figure 2.1 has two slices:

Slice 1: t1

Slice 2: t2, t3

2.1.2 Petri Nets with Markings

A Petri Net can contain *tokens*. Tokens are depicted graphically by indistinguishable dots (•), and reside in places. The existence of one or more tokens represents either the availability of the resource, or the fulfillment of the condition, or the number of items of information in the buffer. The travel of tokens through the net is controlled by the transitions. A *marking* of a Petri Net is a mapping M that assigns a non negative integer (the number of tokens) to each place.

Consider the Petri Net in Fig. 2.2.

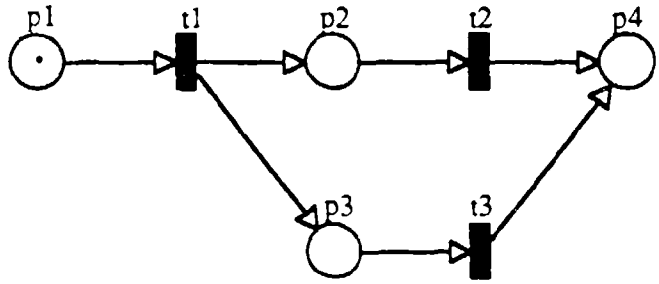


Fig. 2.2 Petri Net PN1 with Marking

The marking is: M(p1) = 1; M(p2) = M(p3) = M(p4) = 0.

Definition 2.7

A transition is *enabled* by a marking, if and only if all of its input places contain at least one token.

On Fig. 2.2, t1 is enabled. All the conditions to be satisfied are fulfiiled.

Definition 2.8

An enabled transition can *fire*. The firing of the transition corresponds to the execution of the

process represented by the transition or the algorithm contained in it. The dynamical behavior of the system is embedded in the movement of the tokens; when the firing takes place, a new marking is obtained by *removing* a token from each input place and *adding* a token to each output place.

In Fig. 2.2, if t1 fires, then the resulting marking is shown in Fig. 2.3.

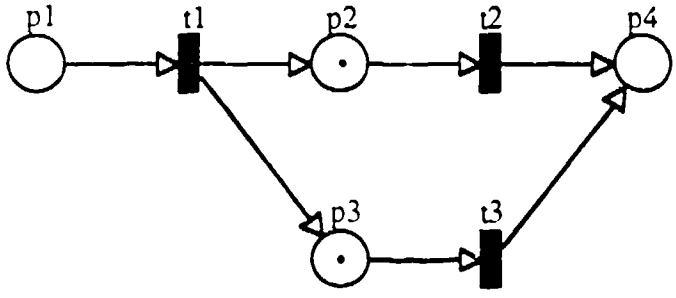


Fig. 2.3 Petri Net PN1 after Firing

Transitions t3 and t2 are now enabled. If they both fire, the new marking is shown in Figure 2.4.

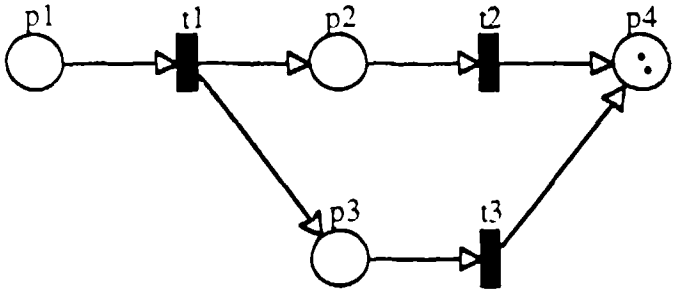


Fig. 2.4 Petri Net PN1 after second Firing

Remark: A transition may fire concurrently more than one token, i.e., a process may handle several tasks simultaneously. Each firing of a transition is thus characterized by an integer k, the *firing pattern* of the transition. A transition can fire according to the firing pattern k, if and only if all of its input places have at least k tokens. When the firing takes place, k tokens are removed from each input place, and k tokens are added to each output place. The firing pattern is 0 if a transition does not fire.

2.1.3 Linear Algebraic Approach

So far, Petri Nets have been described as graphs. An alternative and similar approach can be developed using linear algebra with integer coefficients (Memmi and Roucairol, 1980).

Definition 2.9

A Petri Net with n places and m transitions can be represented by a $n \times m$ matrix C, named the *Incidence Matrix*. The rows correspond to places, the columns correspond to transitions. Its elements are:

$$C_{i,j} = O(t_j, p_i) - I(p_i, t_j), \qquad 1 \le i \le n, 1 \le j \le m. \tag{2.1}$$

- $C_{i,j} = 1$ if there is a directed arc from the j-th transition to the i-th place. 1 indicates that the firing of the j-th transition adds one token to the i-th place.
- $C_{i,j} = -1$ if there is a directed arc from the i-th place to the j-th transition. -1 indicates that the firing of the j-th transition removes one token from the i-th place.
- $C_{i,j} = 0$ if there is no arc from the j-th transition to the i-th place.

For example, the incidence matrix of the net on Fig. 2.1 is:

$$C(PN1) = \begin{array}{c} \begin{matrix} t1 & t2 & t3 \end{matrix} \\ \begin{bmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} p1 \\ p2 \\ p3 \\ p4 \end{matrix}$$

Properties

- The marking of a net can be represented by a $n \times 1$ vector *M*, where $M_i = M(p_i)$. The i-th entry corresponds to the number of tokens in the i-th place.
- The firing pattern of the net can be represented by an $m \times 1$ firing vector *F*, where F_j is the firing pattern of the i-th transition.
- Given an incidence matrix C, an initial marking M, and a firing pattern F, the new marking M' is:

$$M' = M + C * F. \tag{2.2}$$

The matrix equation that corresponds to the firing of the net on Fig. 2.3 is:

$$M' = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 2 \end{bmatrix}$$

M C F

2.1.4 Petri Nets with Switches

For the modeling of decision making organizations, switches have been introduced as an extension of the Petri Net theory to take into account the possible alternatives (Tabak and Levis, 1985). A switch is a particular transition with multiple output places. When a switch fires, only one of its output places can receive a token. This output place which receives the token is chosen according to certain decision rules associated with the switch. These decision rules can be either deterministic (the output place is a function of the input), or stochastic (a probability distribution over the set of possible output places is defined). Figure 2.5 depicts a Petri Net, PN2, with a switch s1.

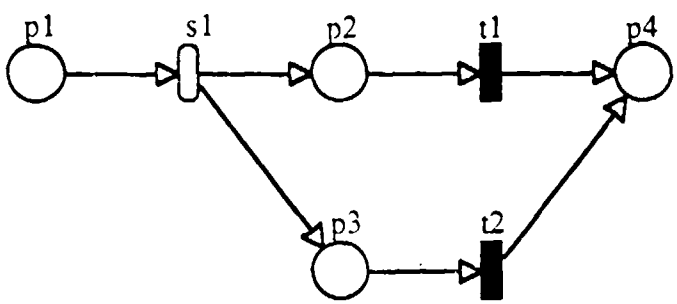


Figure 2.5 Petri Net PN2 with a Switch

A pure Petri Net with switches can be represented also with an incidence matrix. Switches are considered to be transitions and appear on the last columns of the matrix. Nothing about the decision rules of the switch is contained in the matrix representation. The incidence matrix of the Petri net PN2 is :

$$C(PN2) = \begin{array}{c} \begin{matrix} t1 & t2 & s1 \end{matrix} \\ \begin{bmatrix} 0 & 0 & -1 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} p1 \\ p2 \\ p3 \\ p4 \end{matrix}$$

For the modeling of decisionmaking organizations and of distributed intelligent systems, a Petri Net is a formal model of information flow. Tokens can be considered as symbolic information carriers; places are the nodes where tokens can stand without being modified; transitions and switches are events that perform a transformation on the information: it can be a transmission, a computation or a decision; switches are particular types of events that transform input information according to a certain decision rule.

2.2 COLORED PETRI NETS

Ordinary Petri Nets present some problems when modeling and analyzing a distributed system. The first one is that a Petri Net representation of a real system can become very large since it can not take into account some symmetries or repetitiveness displayed by the system. The second problem is the use of switches for modeling alternate courses of actions. When examining the dynamical behavior of such a system, coordination of the settings of several switches is necessary to avoid the creation of deadlocks. Monguillet (1987) describes this problem extensively. Finally, the lack of differentiation of tokens in Ordinary Petri Nets is an important drawback since in the modeling of a system there is a need to know what a token represents specifically when it is present in a given place.

In order to improve the modeling and analytical power of Ordinary Petri Nets, extensions called High Level Nets (Genrich and Lautenbach, 1981) have been devised. Two major models have been developed, Predicate Transition Nets (Genrich, 1987) and Colored Petri Nets (Jensen, 1987), which are used in this report. These models are based on common ideas:

- The tokens can be differentiated. They have an identity (color).
- The identity describes some information about the physical meaning of the token. If a token models a communications message, its identity may be the pair (Sender, Receiver), which describes the source and the destination of the message.
- They provide a way for describing the logic that drives the control process of the transitions.

2.2.1 Definitions

Let us illustrate these notions by introducing Colored Petri Nets, as defined in Jensen (1987), which are used further in this report to model the requirements of a system in different modes of functioning.

Definition 2.10

A Colored Petri Net (CPN) is given by (P, T, $C(t)_{t \text{ in } T}$, $C(p)_{p \text{ in } P}$, $M(p)_{p \text{ in } P}$, I, O):

- *P*, a set of places. As in Ordinary Petri Nets, a place models a resource, a buffer, or a condition, and is depicted by a circle node.
- *T*, a set of transitions. As in Ordinary Petri Nets, a transition models a process, an event, or an algorithm, and is depicted by a bar node.
- Each transition t has attached to it a finite set of *occurrence-colors* with π_t elements: $C(t) = \{ C(t)1, ..., C(t)\pi_t\}$. Each occurrence color corresponds to one firing mode: to one pattern of behavior of the process. In the case of Ordinary Petri Nets, when a process offers s > 1 courses of action, it is modeled by a switch with s branches. In the case of Colored Petri Nets, every process is modeled by a transition and the set C(t) keeps track of the alternative courses of action.
- Each place has attached to it a finite set of *token-colors* with π_p elements. $C(p) = \{C(p)1, ..,C(p)\pi_p\}$. Each token color corresponds to one type of information content attached to a token. Only tokens that have their color in C(p) can be placed in p and one place can simultaneously contain several tokens with the same color. Conversely, one place can contain tokens of different types, provided that their color belongs to C(p).
- The marking of a place is a $\pi_p \times 1$ vector M(p). $M(p) = [\beta_1, .., \beta_i, ..., \beta_{\pi p}]$, where β_i indicates that p contains β_i tokens of color C(p)i.
- *I(p,t) is a $\pi_p \times \pi_t$ matrix* for any transition t and any place p. The rows correspond to the elements of the set of token colors, C(p), that is the colors of the tokens that can be put in the place p. The columns correspond to the elements of the set of occurrence colors of t, C(t), the alternative courses of actions. $I(p,t)_{i,j}$ describes the number of tokens of color C(p)i that are removed from the place p, when the transition t fires according to the firing mode j. If some entries of I(p, t) are non null, an arc is drawn from the place p to the transition t in the graphical representation of the Colored Petri Net. This arc is annotated by the matrix I(p, t).
- *O(p,t) is a $\pi_p \times \pi_t$ matrix* for any transition t and any place p.

The rows correspond to the elements of the set of token colors, C(p), the colors of the tokens that can be put in the place p. The columns correspond to the elements of the set of occurrence colors of t, C(t), the alternative courses of actions.
$O(p,t)_{ij}$ describes the number of tokens of color C(p)i that are put in the place p, when the transition t fires according to the firing mode j. If some entries of O(p, t) are non null, an arc is drawn from the place p to the transition t in the graphical representation of the Colored Petri Net. This arc is annotated by the matrix O(p, t).

2.2.2 Examples

Let us describe two examples to understand the concepts of Colored Nets. These examples are from Demaël (1989).

Example 1: Product Sorting

Consider a machine at the end of an assembly line. This assembly line produce the products of type A, B and C. The machine sorts these products and has to put them in appropriate boxes according to their types. This job can be represented by a Colored Petri Net in Figure 2.6.

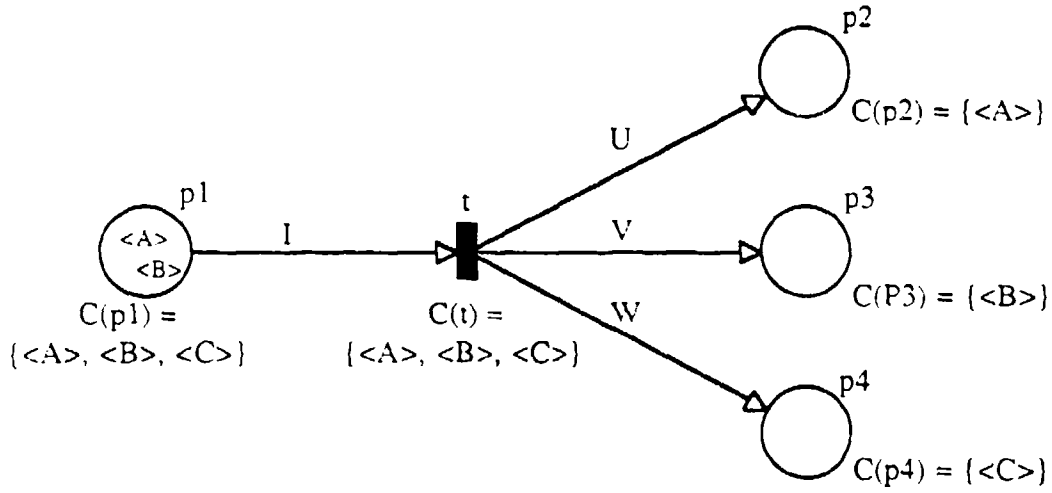


Fig. 2.6 Product Sorting

In this example, places can contains tokens, which represent products. It is thus very natural to assume that the color of the tokens belong to {<A>, <B>, <C>}, the types of products.

Place p1 stands for the products to be sorted. Its color set is {<A>, <B>, <C>}, as this place can contain any combination of products to be sorted. The initial conditions depicted on Figure 2.6 are that two products, one A and one B have to be sorted.

Place p2 models A's box, which contains exclusively the products of type A, C(p2) = {<A>}. Similarly, p3 is the box for products of type B, and C(p3) = {<B>}. Finally, the place p4 is the box for products of type C, and C(p4) = {<C>}.

The transition t models sorting. The machine has three courses of actions, which have also been labeled by <A>, <B>, and <C>. <A> models the fact that the machine is sorting some products of type A, <B> models sorting products of type B, and <C> models sorting products of type C.

The arcs of the CPN have been annotated by the matrices I, U, V, W, where I is

$$I = \begin{array}{c} \begin{matrix} \langle A\rangle & \langle B\rangle & \langle C\rangle \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} \langle A\rangle \\ \langle B\rangle \\ \langle C\rangle \end{matrix}$$

I indicates that the machine takes only one product A when he sorts products of type A, that it takes one product B when it sorts products of type B, and that it takes only one product C when it sorts products of type C. The other matrices that annotate the arcs are:

$$U = \begin{matrix} (A) & (B) & (C) \\ [1 & 0 & 0] \end{matrix} \qquad V = \begin{matrix} (A) & (B) & (C) \\ [0 & 1 & 0] \end{matrix} \qquad W = \begin{matrix} (A) & (B) & (C) \\ [0 & 0 & 1]. \end{matrix}$$

The matrix U indicates that one product is put into the box of products of type A only if the machine sorts products of type A. V indicates that one product is put into the box of products of type B only if the machine sorts products of type B. W indicates that one product is put into the box of products of type C only if the machine sorts products of type C.

This example is simple, because the token-colors are intuitive, and because the firing modes of t correspond exactly to the token-colors. The next example describes a Colored Petri Net that is less obvious.

Example 2

Figure 2.7 represents another Colored Petri Net.
The set of places is P = {p1, p2, p3, p4, p5}.
The set of transitions is T = {t1, t2, t3}.
The set of token-colors for p1 and p2 is A = {a1, a2}.
The set of token-colors for p3 and p4 is B = { b1, b2}.
The set of token-colors for p5 is C = {c1, c2, c3}.
The transitions t1 and t2 have two firing modes, which are labeled 1 and 2.
The transition t3 has three firing modes, which are labeled 1, 2, and 3.

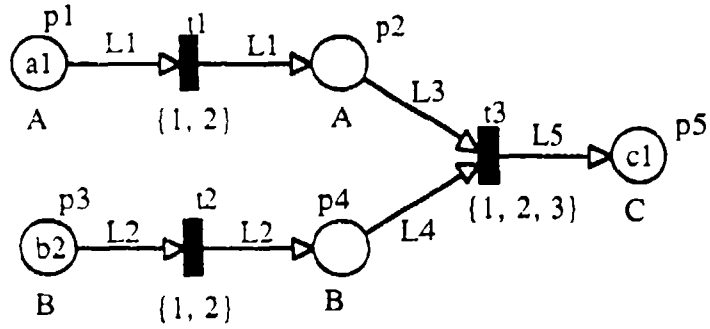


Fig. 2.7 Colored Petri Net

where the matrices that annotate the arcs are

$$L1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix},\quad L2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\quad L3 = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix},\quad L4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix},\quad L5 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- L1 indicates that the first firing mode of t1 (first column of L1) removes one token of color a1 from p1, and places one token in p2. The second firing mode (second column) removes one token of color a1 and one token of color a2, and places both in p2 .
- L2 indicates that the first firing mode of t2 removes one token of color b1 from p3, and places it in p4. The second firing mode removes a token of color b2, and places it in p4.
- L3 indicates that the first two firing modes of t3 remove one token of color a1 from p2. The third firing mode removes one token of color a2.

- L4 indicates that the first firing mode of t3 removes one token of color b1 from p4. The other two firing modes remove one token of color b2.
- L5 indicates that all firing modes of t3 place one token of color c1 in p5.

This example shows how to deal with tokens of different colors and how they can be combined together. For example, the first firing mode of t3 shows that a token of color a1 has to be removed from p2 and a token of color b1 from p4 to produce a token of color c1 in p5 with a1, b1 and c1 belonging to different sets of token-colors.

2.2.3 Firing Rules

The firing rules for Colored Nets are similar to those of Ordinary Petri Nets. The firing mode C(t)i is *enabled* for the transition t if and only if each input place of t contains at least the colored tokens that are indicated by the i-th column of I(p,t). An enabled transition can *fire* if at least one of its firing modes is enabled. If the transition t fires according to the firing mode C(t)i, the colored tokens that are indicated by the i-th column of I(p,t) are removed from each input place p. For each output place p', colored tokens that correspond to the i-th column of O(p',t) are added to the place p'.

One transition may fire concurrently according to several modes, i.e., a process may be able to handle tasks of a different nature at the same time. Within each category, i.e., given one firing mode, several tasks of the same nature may be processed concurrently. The *firing pattern* of a transition in a Colored Petri Net is thus given by a $n_t \times 1$ vector F_t, where $[F_t]_i$ describes the number of concurrent activations of the i-th firing mode. If the firing pattern of the transition is F_t, then the combination of colored tokens indicated by the i-th column of I(p, t) is removed $[F_t]_i$ times from each input place p. Similarly, the combination of colored tokens indicated by the i-th column of I(p', t) is added $[F_t]_i$ times to every output place p'.

In the CPN of Fig 2.7, only the first mode of t1 and the second mode of t2 are enabled:

- The input place of t1, p1, contains only one token of color a1. L1, the annotation of the adjacent arc is:

$$L1 = \begin{matrix} 1 & 2 \\ \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} & \begin{matrix} a1 \\ a2 \end{matrix} \end{matrix}$$

The firing mode 1 removes one token of color a1. The initial marking of p1 enables this firing mode. This is not the case for firing mode 2, because p1 does not contain a token of

color a2. Finally, as p1 contains one and only one token of color a1, t1 can process at most one task a1, and $F(t)_1$ is either 0 or 1.

- The input place of t2, p3, contains only one token of color b2. L2, the annotation of the adjacent arc is:

$$L2 = \begin{array}{c} \begin{matrix} 1 & 2 \end{matrix} \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{array} \begin{matrix} b1 \\ b2 \end{matrix}$$

Firing mode 1 removes one token of color b1. The marking of p3 does not enable this firing mode. This is not the case for firing mode 2, because p3 does contain a token of color b2. Finally, as p3 contains one and only one token of color b2, t2 can process at most one task b2, and $F(t)_2$ is either 0 or 1.

If t1 fires according to the firing mode 1, and t2 according to its firing mode 2, the marking of the net is changed into the marking of Figure 2.8.

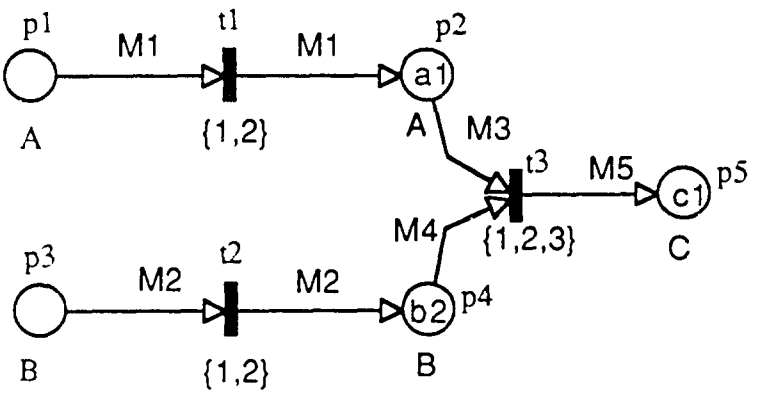


Fig. 2.8 Colored Petri Net after Firing

2.2.4 Incidence Matrix

A Colored Petri Net with n places and m transitions can be represented by a n × m block matrix C: the *Incidence Matrix*. The entries of the matrix are themselves matrices.

The rows correspond to places, the columns correspond to transitions.

- $C_{i,j} = O(t,p)$, if there is a directed arc from the j-th transition to the i-th place. O(t,p) indicates the colored tokens that can be added, as determined by the firing modes.

- $C_{i,j}$ = -I(t,p), if there is a directed arc from the i-th place to the j-th transition. -I(t,p) indicates the colored tokens that can be removed, as determined by the firing modes.
- $C_{i,j}$ = Null matrix, if there are no arcs.

The marking of a net can be represented by a n×1 block vector *M*, where M_i = M(pi). The i-th entry corresponds to the colored tokens in the i-th place.The i-th entry is thus a column with one row for each element of C(pi).

Given an incidence matrix C, an initial marking M, and a firing pattern F, the new marking is:

$$M' = M + C * F \qquad (2.3)$$

The Colored Petri Net of Fig. 2.7 has the following Incidence matrix:

$$C = \begin{bmatrix} -L1 & 0 & 0 \\ L1 & 0 & -L3 \\ 0 & -L2 & 0 \\ 0 & L2 & -L4 \\ 0 & 0 & L5 \end{bmatrix}$$

The initial marking is given by:

$$M = \begin{bmatrix} M(p1) \\ M(p2) \\ M(p3) \\ M(p4) \\ M(p5) \end{bmatrix}, \text{ with } M(p1) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, M(p3) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, M(p2) = M(p4) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, M(p5) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

The place p1 contains only one token of color a1, p3 contains only one token of color b2, p2 and p4 contain no tokens, and p5 contains only one token of color c1. Finally, the firing pattern of Fig. 2.8 corresponds to

$$F = \begin{bmatrix} F1 \\ F2 \\ F3 \end{bmatrix}, \text{ with } F1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, F2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, F3 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

By applying (2.3) the new marking M' is obtained:

$$M' = \begin{bmatrix} M'(p1) \\ M'(p2) \\ M'(p3) \\ M'(p4) \\ M'(p5) \end{bmatrix}, \text{ with } M'(p1) = M'(p3) = \begin{bmatrix} 0 \\ \\ 0 \end{bmatrix}, M'(p2) = \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix}, M'(p4) = \begin{bmatrix} 0 \\ \\ 1 \end{bmatrix}, M'(p5) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

This chapter introduced basic concepts of Ordinary Petri Net theory and Colored Petri Net theory. Only the notions that are of relevance in the subsequent chapters have been defined.

CHAPTER 3

THE CUBE TOOL METHODOLOGY

3.0 INTRODUCTION

C3I (Command, Control, Communication and Intelligence) systems are distributed systems in the sense that they are geographically dispersed. They involve a time-dimension which can be evolutionary and cover a multiplicity of different operational domains. Such systems must allow for technical and functional reconfiguration and an evolutionary implementation. These systems associate operators, software, hardware, facilities and procedures which are organized to satisfy a set of specific needs and are subject to technical and operational constraints. Therefore, a method for designing C3I systems has to take into account all the human and technical aspects.

Cube Tool (Tournes, 1988) is a methodology developed at THOMSON-CSF in France for the design and the analysis of C3I systems. The methodology allows for (1) the qualitative and quantitative design of the architecture of C3I systems; (2) the determination of the characteristics of the elements, which are known also as the attributes or parameters of the system; and (3) the definition of the general plan for realization. The Cube Tool covers the application domains which are common to all C3I systems: communication, information processing, information storage, supervision/management, and man/machine interface. The application of Cube Tool to the design and the analysis of a system is done in four steps, as shown in Figure 3.1.

- Identification of the system Functions and of the different Actors or resources (personnel and hardware/software) involved,
- Functional Analysis for the determination of the processing and information exchanges for each function,
- Quantitative Evaluation of Automated Data Processing (ADP) and communication loads in workstations,
- Consideration of Alternative Architectures through the allocation of functions to different sites.

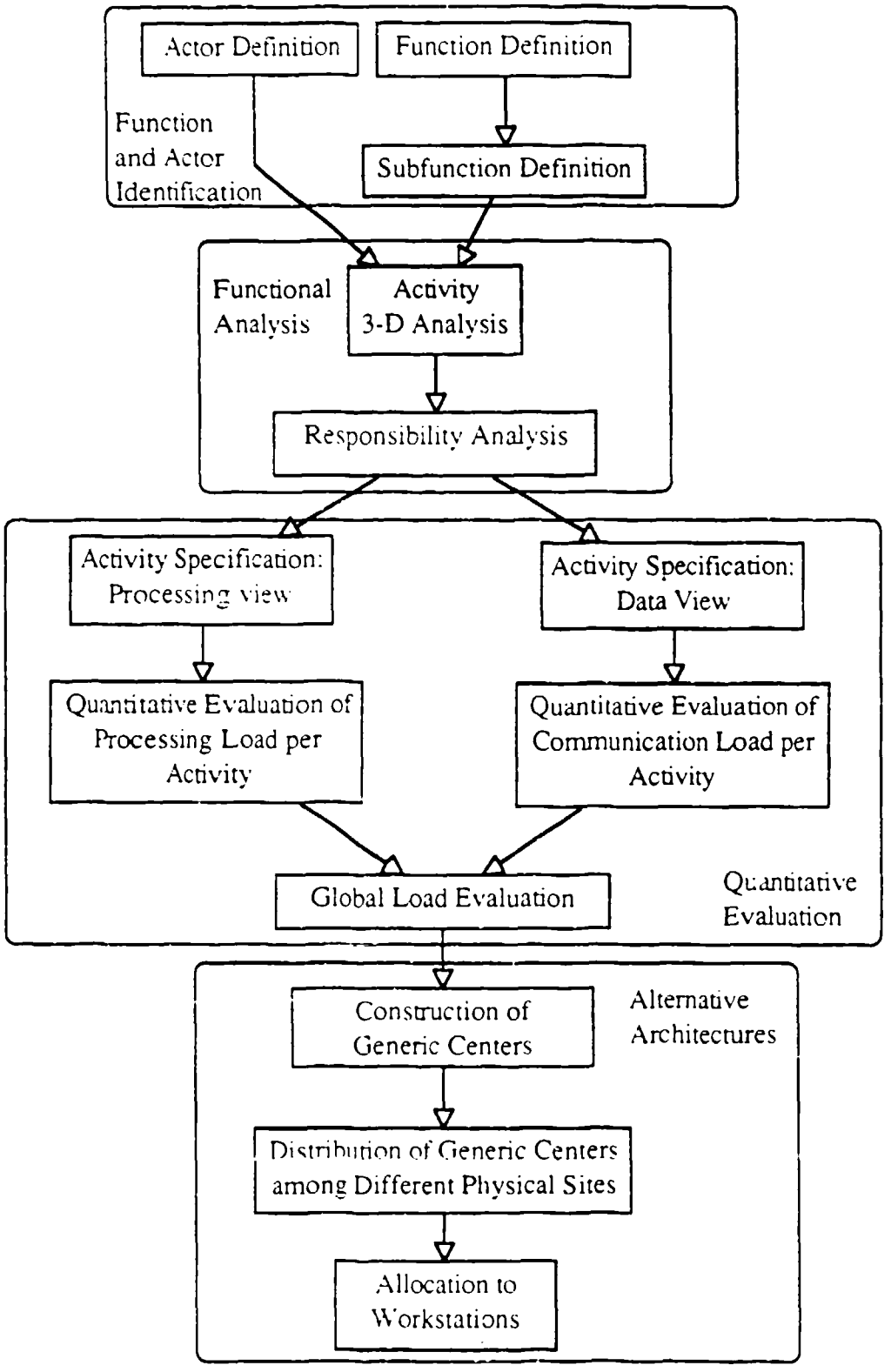


Figure 3.1 Methodology Flow Chart

3.1 STEP 1: SYSTEM FUNCTION AND ACTOR IDENTIFICATION

The first stage of Cube Tool consists of identifying the functions of the system to be designed. Simultaneously, the resources needed for the execution of these functions are defined. They consist of personnel and hardware/software entities such as databases or decision aids and are referred to as Actors. At this stage, the designer must find out the user needs, the type of missions the system will have to accomplish, and the personnel and types of hardware and software which will be used. This process requires intensive interviews with the user to determine exactly what the range of operations of the system will be. The missions that the system is expected to carry out are determined and are used as the basis for the identification of the global tasks that must be executed for the fulfillment of a mission. These global tasks are the system Functions. For example, a system for planning an air interdiction mission will have as functions the determination of the status of allied forces, weather projection, threat assessment, strike assessment, intelligence report processing, target prioritization and development, weapon system availability, etc.

Then, each system function can be decomposed in subfunctions. The processing tasks are differentiated from the transmission tasks. A processing task only involves the processing of data received by an actor in charge of creating or inferring new information. Transmission tasks only involve the communication of information between two different actors without any alteration in the content. A function can be considered to be an interleaved sequence of processing and communication tasks, a subfunction can be defined as a single pair consisting of a process task and a communication task. The execution of a function will require the sequential execution of its subfunctions.

3.2 STEP 2: FUNCTIONAL ANALYSIS

In a second stage, a functional analysis is performed for each function in a three dimensional space, as shown on Figure 3.2. The three axes of interest are :

- Functions: These are the processes which have to be executed for the fulfillment of the mission.
- Actors or Hierarchical Levels: These are the personnel and the hardware and software nodes responsible for executing the different tasks. Personnel are layered in hierarchical levels and are most of the time specialized per functional domain
- Time: This axis shows on the same scale the execution time of the functions, their frequency

and their sequence.

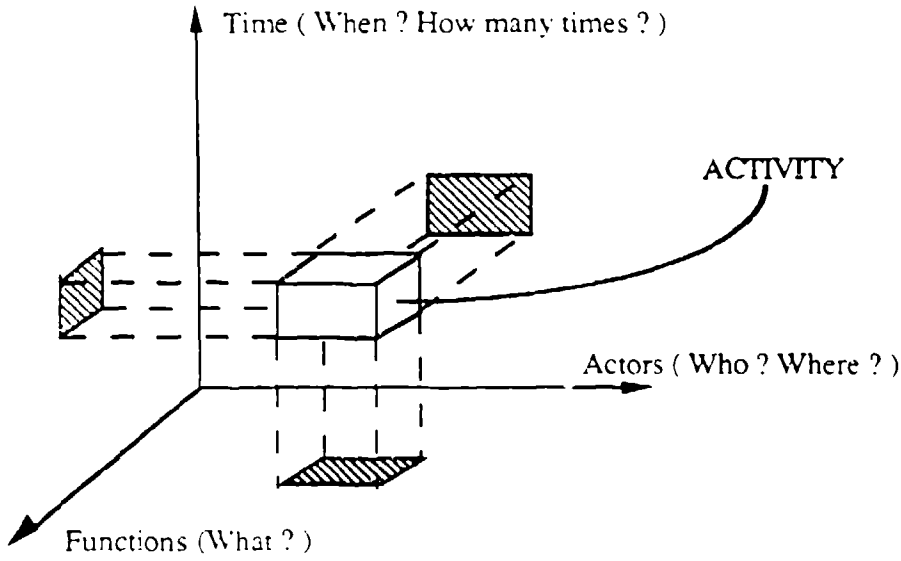


Figure 3.2 Three-Dimensional Functional Analysis

In this framework, subfunctions are defined as a collection of activities with their interrelated information exchanges. An activity is defined as a process which supports a given system subfunction and which is performed by a single actor or hierarchical level without major interruption. Therefore, activities can be part of a processing task, a communication task, or contain elements of both. An activity is thus represented as a cell in the three dimensional space. The functional analysis is performed by considering the projection of the cells on each of the three planes defined by the axes taken two by two, as shown on Figure 3.2. These analysis planes are shown on Figure 3.3. These are:

- Responsibilities Plane (Functions / Actors): This plane shows which actor is in charge of a set of specific activities.
- Sequences Plane (Functions / Time): This plane shows when and how many times an activity will be executed.
- Actions Plane (Time / Actors): The plane of actions shows when actors are busy performing some activity.

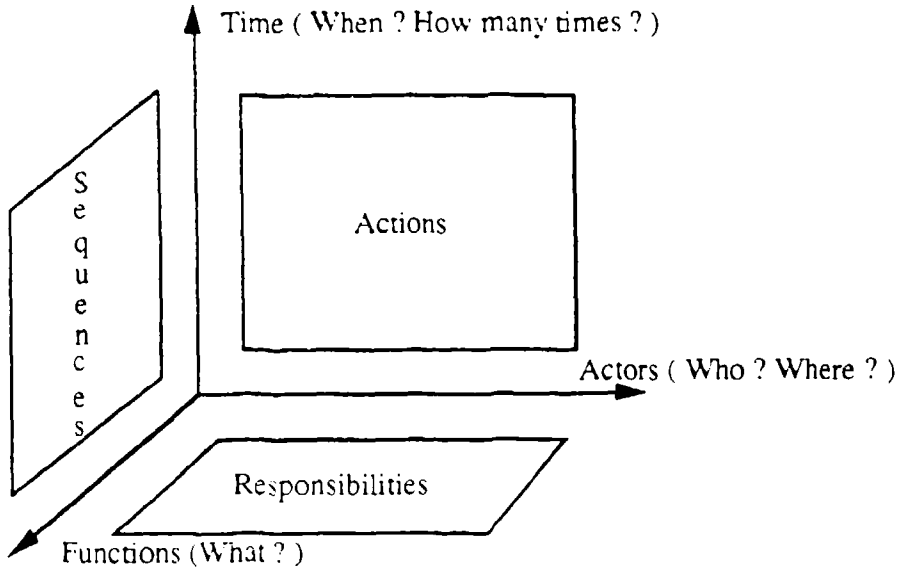


Figure 3.3 The Three Analysis Planes

On each of the analysis planes the analysis is done according two perspectives:

- The Processing View, which focuses on the tasks which transform information into new information
- The Data View, which focuses on the tasks which transmits information without modifying them.

To perform the functional analysis, activities are differentiated according to the type of processing they represent and are called *roles*. The roles considered by the method are:

- *Elaborate* (**E**): generate or transform information.
- *Acknowledge* (**A**): receive an order important enough to warrant the generation of an acknowledgement.
- *Check* (**C**): receive a report in response to an order or request previously generated.
- *Warn* (**W**): receive information which does not require taking any measures in the current mode of operation.
- *Monitor* (**M**): receive information on system status and operations in support of command, control, and communication resources management.
- *Monitor Locally* (**L**): same as M but on a local basis
- *Secure* (**H**): exchange of secure data such as encryption keys, access keys and certification mechanisms of users' trustworthiness.

The main analysis is performed in the responsibility plane. The roles which are used most and are the only ones considered for the requirements specification are E, A, C and W. The responsibility plane is constructed by allocating the roles for each subfunction to the different actors. This allocation must verify the following rules:

- Roles E are assigned using common sense or in accordance with the user's wishes. Some actors are more qualified to execute the processing part of a certain subfunction and expertise can be used as a criterion for the allocation of roles E to the different actors.
- There is one and only one role E per subfunction, which means that there is one and only one role E per row in the array of responsibilities.
- Then, the different roles E have to be connected with communication interfaces to make possible the distributed execution of the function. Communications between two actors are represented with pairs E→X where X = A, W or C:
    - E→A corresponds to an order sent from the actor performing the role E to the actor performing the role A
    - E→C corresponds to a report in response to an order previously generated sent from the actor performing the role E to the actor performing the role C. This means that if the pair E→A exists on a row, the pair C←E has to be present in a row below as shown on Figure 3.4.

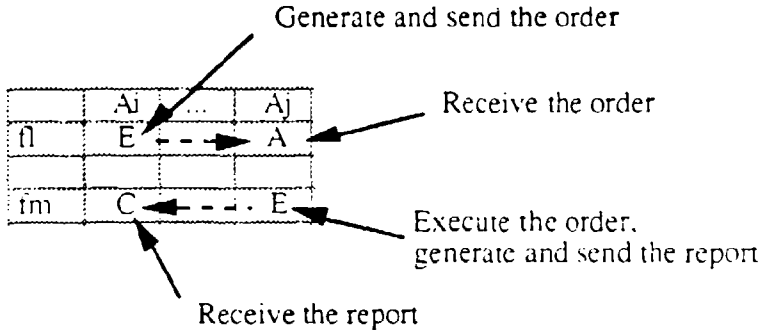


Figure 3.4 Order-Execution-Report Representation on the Responsibilities Plane

- E→W corresponds to the transmission of a warning, i.e., that some event has happened and that certain procedures will have to take place. This communication takes place usually from an actor of higher hierarchical level to an actor of lower hierarchical level but the notion can be extended to the exchange of data or information regardless of the hierarchical level.

This is illustrated in the example shown on Table 3.1.

Table 3.1: Responsibilities for a Function with Six Subfunctions Performed by Four Actors

	actor 1	actor 2	actor 3	actor 4
subfunction 1	E	A	W	
subfunction 2	C	E	A	W
subfunction 3		C	E	A
subfunction 4			C	E
subfunction 5		C	E	
subfunction 6	C	E	W	

Explicit exchanges take place across columns, between activities contributing to the execution of the same subfunction (i.e., on same row). Implicit exchanges occur from row to row between activities performed by a single actor. The interesting aspect of this methodology is that several configurations, differing as to the resources used or reflecting variations in operational needs, can be represented in a consistent manner. This makes it possible to define different thresholds of responsibilities in different modes (normal mode or emergency modes) and to point out how the reallocation of the tasks has to be made among the available actors when the system switches from one mode to another.

3.3 STEP 3: QUANTITATIVE EVALUATION OF ADP AND COMMUNICATION LOADS IN WORKSTATIONS

The third step, which is the quantitative evaluation of ADP and communication load in workstations, is performed through an activity specification according to the processing view and the data view. From these specifications, the quantification can be made.

3.3.1 Activity Specification According to the Processing View

The activity specification is done only for the role E. For the other roles (A, C, W, L, M and H) a generic specification is assumed for each role and used independently on the subfunction to which they belong. For activities of type E, each activity is defined using a pseudo-code formalism close to Pascal or ADA. As shown in the example displayed on Figure 3.5, this formalism has an indented block structure with an indication of the number of times each block loop is executed. Primitives which define the nature of the information transformed are used and

gathered in a dictionary. A primitive is decomposed in four parts:

(1) an *operation* on the information such as retrieve, update, send, display, call ...

(2) the first Information module which consists of the type of information (Message, Program, Graphic, Database,...) and the information name which references the data or the processing used throughout the system.

(3) the direction (to, from, with, on,...)

(4) the second information module which has the same structure as the first one.

Examples of primitives are:

Update Database **Enemy_position** from alphanumeric MMI **Mission_Report**

Call subroutine **Mission_Preparation** with **Enemy_Position** Data

The information names defined in the two information modules are used to check the consistency of processing and of communication. In the example above, **Enemy_position** is the name of the database recording the position of the enemy, **Mission_report** is a display and **Mission_preparation** is a processing reference.

```
BEGIN GENERAL TASKING
  FOR N=1 .. N REQUEST
     DISPLAY ALPHANU MIS-PREP FROM MSG TARGT-REQ
     UPD DB TARG-REQ-DATA FROM DISP ALPHANU  MIS-PREP
  END FOR
  FOR N=1 .. N REQUEST
     FOR M=1 .. M BASES
        RET DB BASE-POSIT-DATA
        CALL MIS-PLN  WITH TARG-REQ-DATA
           AND BASE-POSIT-DATA
        UPD DB MIS-RESUL  FROM PROGMIS-PLN
     END FOR
  END FOR
  DISP ALPHANU MIS-RESUL
END GENERAL  TASKING
```

Figure 3.5 Example of the Pseudo-code Formalism

To make a quantitative evaluation of the processing load in workstations, an average processing consumption is associated with each primitive invokement. These consumptions are either computed by the summation of elementary instructions or derived from benchmarks A

distinction is made between ordinary processing, expert system scientific processing, man-machine interfacing or communication protocols to ease the allocation of the load to different workstations.

3.3.2 Activity Specification According to the Data view

The activity specification is done by analyzing the ways information is displayed and sent for the incoming and outgoing data, for each activity. As shown on Figure 3.6, these information flows include:

- information exchanges which are the messages,
- data retrieved from the data bases,
- images which might be static, slow-moving, or normal motion images,
- Man Machine Interface (MMI) which might be alphanumeric or graphic.

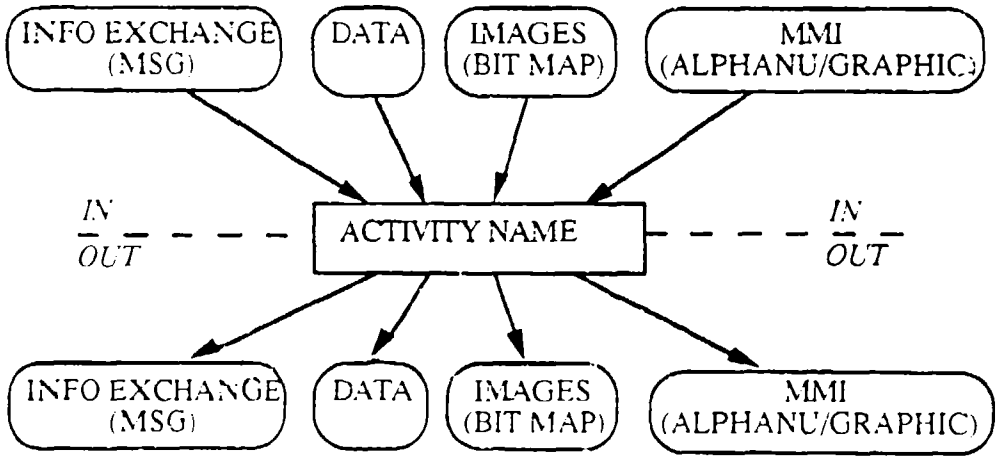


Figure 3.6 Activity Specification from the Data View

To evaluate the communication load for each activity, four types of exchanges are considered:

- voice communications,
- character oriented text,
- bit-oriented messages,
- images.

3.3.3 Global Load Evaluation

The activity specification allows to quantify the load for processing, data storage, man-machine interface manipulation, and communication. Simultaneously, a quantification is made for the maximum response time to determine the minimum processing power threshold. By summing these results for each logical group, which is the set of activities related to a given system function and performed by a single actor, the number and type of workstations, the data processing requirements, the number of database updates and retrievals and the load associated to communication processing and related communication flows can be determined.

3.4 STEP 4: CONSIDERATION OF DIFFERENT ARCHITECTURES

The last stage is the investigation of different possible architectures through the allocation of logical groups to different sites. Generic centers are first defined by gathering the logical groups meant to operate together and which are sufficient to constitute an independent site. This is done in order to check data coherency inside the generic center but also among the different centers. The summation of the related loads is performed at the generic center level and gives an indication of what would be the load of a centralized center supporting all the load of the system related to the set of logical groups handled by this center.

Areas of responsibility and interest are assigned to logical groups. These areas correspond to percentage of geographical involvement that each actor has.

In the next step, generic centers are shared between several physical sites. The load is mapped proportionally to the areas of responsibility and interest assigned to logical groups. Load position factors are applied to take into account the geographical dispersion of the logical groups in the different sites. Simultaneously, different modes of operation of the system are defined. These modes are normal or backup modes which are a reassignment of the activities to different actors in case of emergency. The vertical back-up corresponds to the reallocation of the activities between actors of different hierarchical levels; the horizontal back-up corresponds to the the reallocation of activities between actors of same hierarchical level. A load factor is also defined for each of these modes and used for the allocation of the load to the different sites. These load factors are coefficients with value larger than one which, when multiplied with an estimate of the processing load or communication load, give an indication of the additional loads due to (1) additional communications resulting from the geographical dispersion of the processes and (2) additional effort to switch from one mode of operation to another. This is shown on Figure 3.7.

The load of the generic center is 100%. When the load is distributed among different sites, load factors for geographical dispersion and for possible switching of modes are applied. As a result, the sum of the load of the different sites is larger than 100%.

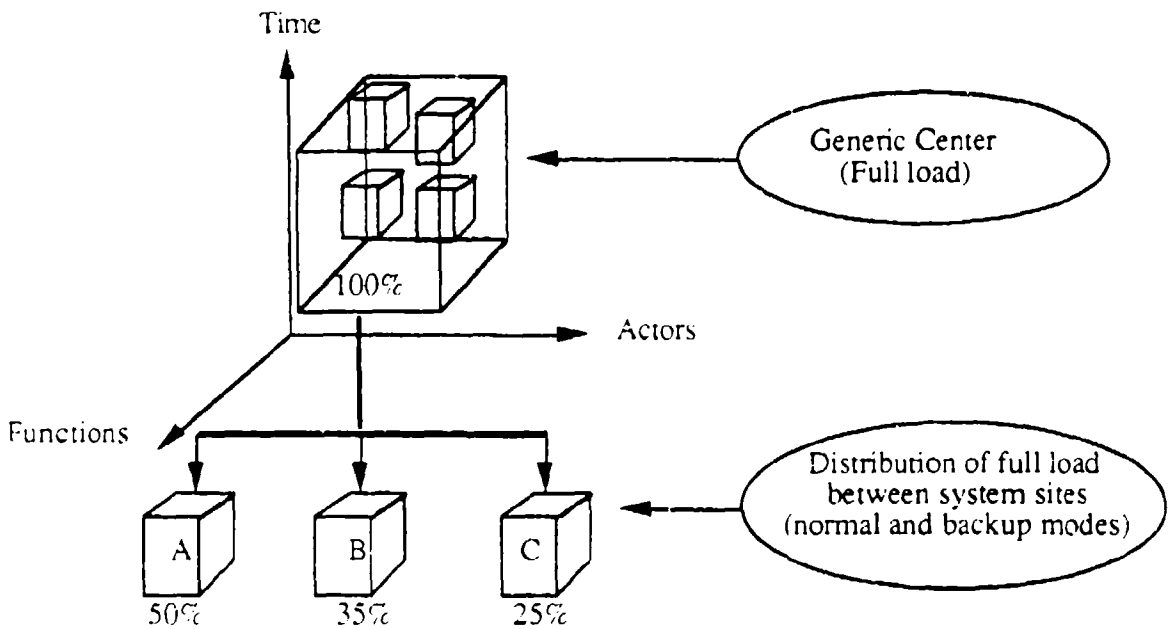


Figure 3.7 Load Sharing

This synthesis gives indications on:

- The processing loads per activity in each site
- The communication loads per activity in each site
- The connectivity matrix for the sites of the system (which site is connected to which site?)
- The coherency for data processing sharing
- The total loads for each site
- The total loads for each system communication networks

The last step is the reallocation of the load within these system sites to the workstations according to the type of processing (normal, expert, scientific) and the security requirements. Different architectures can be generated in this way and the selection of the final one is made according to criteria such as cost or ease of implementation. Other criteria could be taken into account such as the effectiveness of the system

The interesting aspect of this methodology is that it allows to represent in a consistent manner several configurations of the use of the system to allow it to adapt to the availability of the resources or to the variation of the operational needs. This allows to define different thresholds of responsibility in different modes (normal mode or emergency modes) and to point out how the reallocation of the tasks has to be made among the available actors when the system switches from one mode to another. For this report, only the two first steps of the methodology are of interest and are used for the specification of a system requirements. The next chapter shows how to convert the allocation of roles into Petri Nets and how the detailed requirements of a system for a particular mission can be generated.

CHAPTER 4

PETRI NET REPRESENTATION OF REQUIREMENTS

4.0 INTRODUCTION

The requirements of a system are the set of processes which have to take place for the correct execution of a mission. These requirements are scenario-dependent and are most often defined by the set of functions with their sequences and interrelationships. In Valraud and Levis (1989), the requirements are described by a Petri Net in which system functions are represented with transitions and the data produced by these functions, necessary for the execution of subsequent functions, with places. These nodes are connected together to model the relationships among functions and to show what should be their order of execution. This section describes how to develop more detailed requirements of a system which take into account not only the different processes which have to take place, but also the communication exchanges between the different parts of the system.

The Cube Tool can be used to define, for each function, the processes and the communication exchanges among the different actors involved in the execution of that function. The Petri Nets depict graphically these processes and communication exchanges for each function. When these representations are linked together to construct the requirements, a global and consistent graphical representation can be defined that lets the designer or the analyst take advantage of the mathematical framework which underlies Petri Nets.

4.1 REPRESENTING THE RESPONSIBILITIES FOR A FUNCTION WITH PETRI NETS

As we have seen in the previous chapter, the first two steps of Cube Tool result in the definition of the different system functions, their subfunctions, and how the activities constituting these subfunctions are allocated to the different actors of the system. For each system function, the responsibility analysis plane defines the activities performed by the different actors. From this representation, the generation of the equivalent Petri Net representation of the responsibilities for each function is done in three steps.

In the first step, each activity is depicted by a transition. The transitions representing the activities performed by the same actor are aligned horizontally, while the ones representing the

activities belonging to the same subfunction are aligned vertically. In other words, the transpose of the array of responsibilities is obtained and the non-null elements of this array are transformed into transitions, as shown in the Figure 4.1.

function 1

	subfunction 1	subfunction 2	subfunction 3	subfunction 4	subfunction 5	subfunction 6
actor 1	E	C				C
actor 2	A	E	C		C	E
actor 3	W	A	E	C	E	W
actor 4		W	A	E		

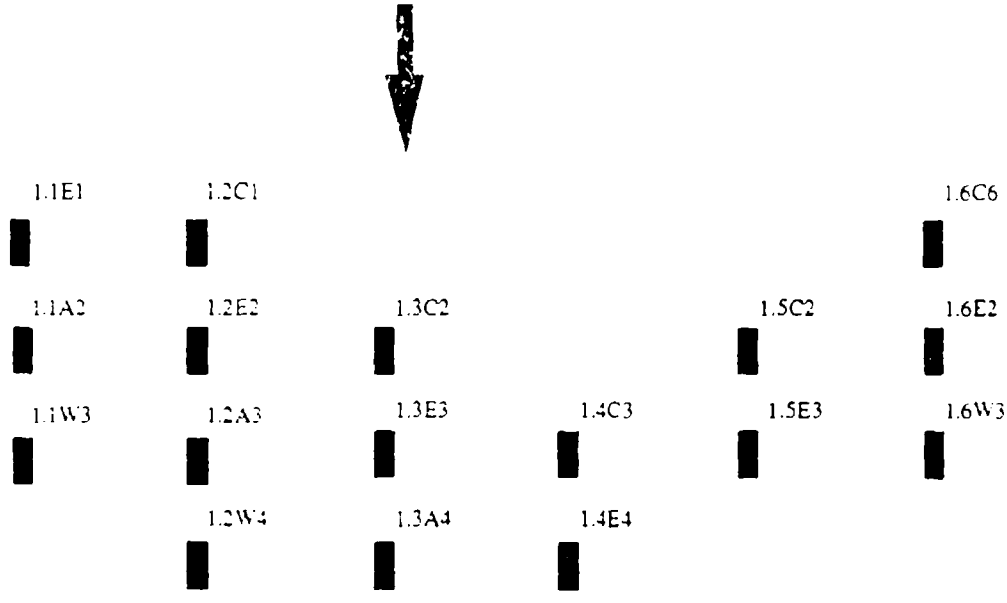


Figure 4.1 Drawing the Transitions Grid

A label is attached to each transition identifying (1) the function, (2) the subfunction to which the represented activity belongs, (3) the type of activity (E, A, C or W) and (4) the actor performing this activity. For example, in Figure 4.1, the label 1.3E3 means that the activity represented by the transition belongs to subfunction 3 of function 1, is of type E, and is performed by actor 3. In the application described in this paper, the subfunctions are not identified by their order of appearance in a function, but by the identification number of the processing they represent throughout the system.

The second step is to add places between the transitions representing the activities performed by a single actor and to connect them. In this way, the implicit information exchanges which take place between the successive activities performed by each actor are modeled. Figure 4.2 shows

the net obtained for the example.

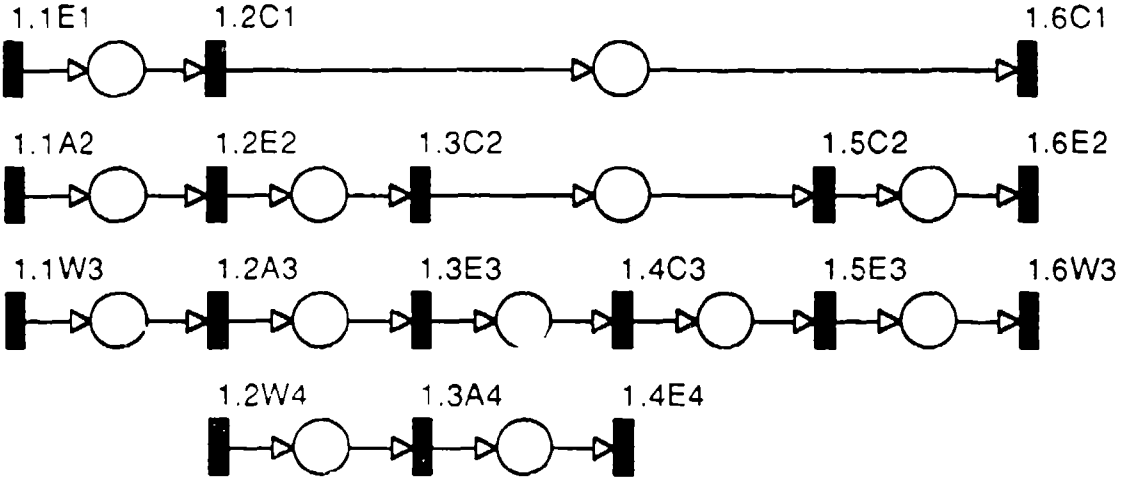


Figure 4.2 Adding Implicit Information Exchanges

The third step consists of adding the information exchanges which take place among the actors for each subfunction. In the Cube Tool methodology, an exchange originates from a role E and ends at a role A, W or C and that there is one and only one role E for each subfunction. Therefore, for each column of the Petri Net representation obtained after the two first steps, the transition representing the role E is identified and is connected to the other transitions of the columns with a connector-place-connector set. Figure 4.3 shows the final net for the example.

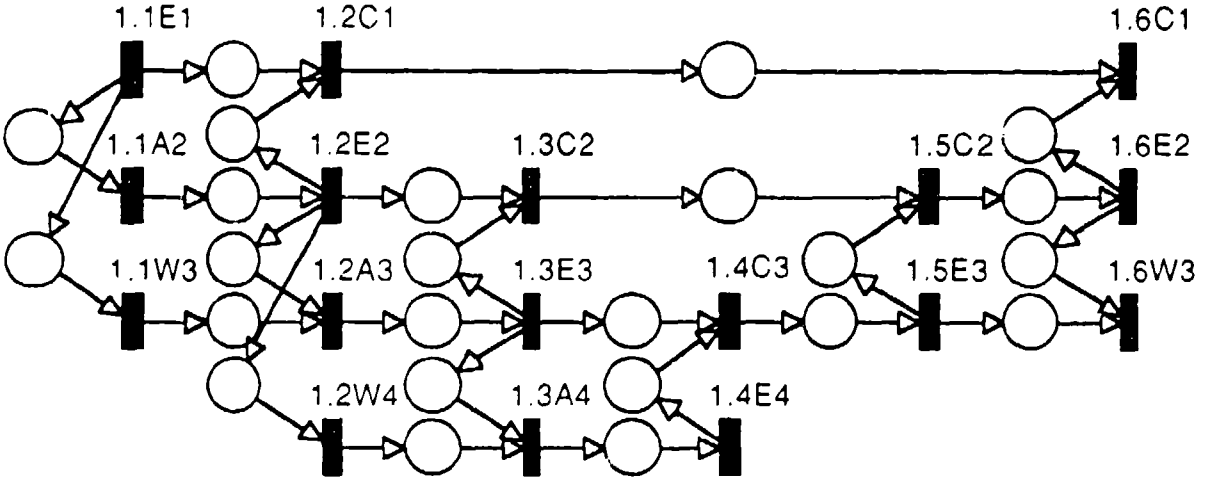


Figure 4.3 Adding Explicit Information Exchanges

4.2 MODELING THE REQUIREMENTS FOR A SCENARIO

The procedures for modeling the detailed requirements for a given scenario is shown on Figure 4.4.

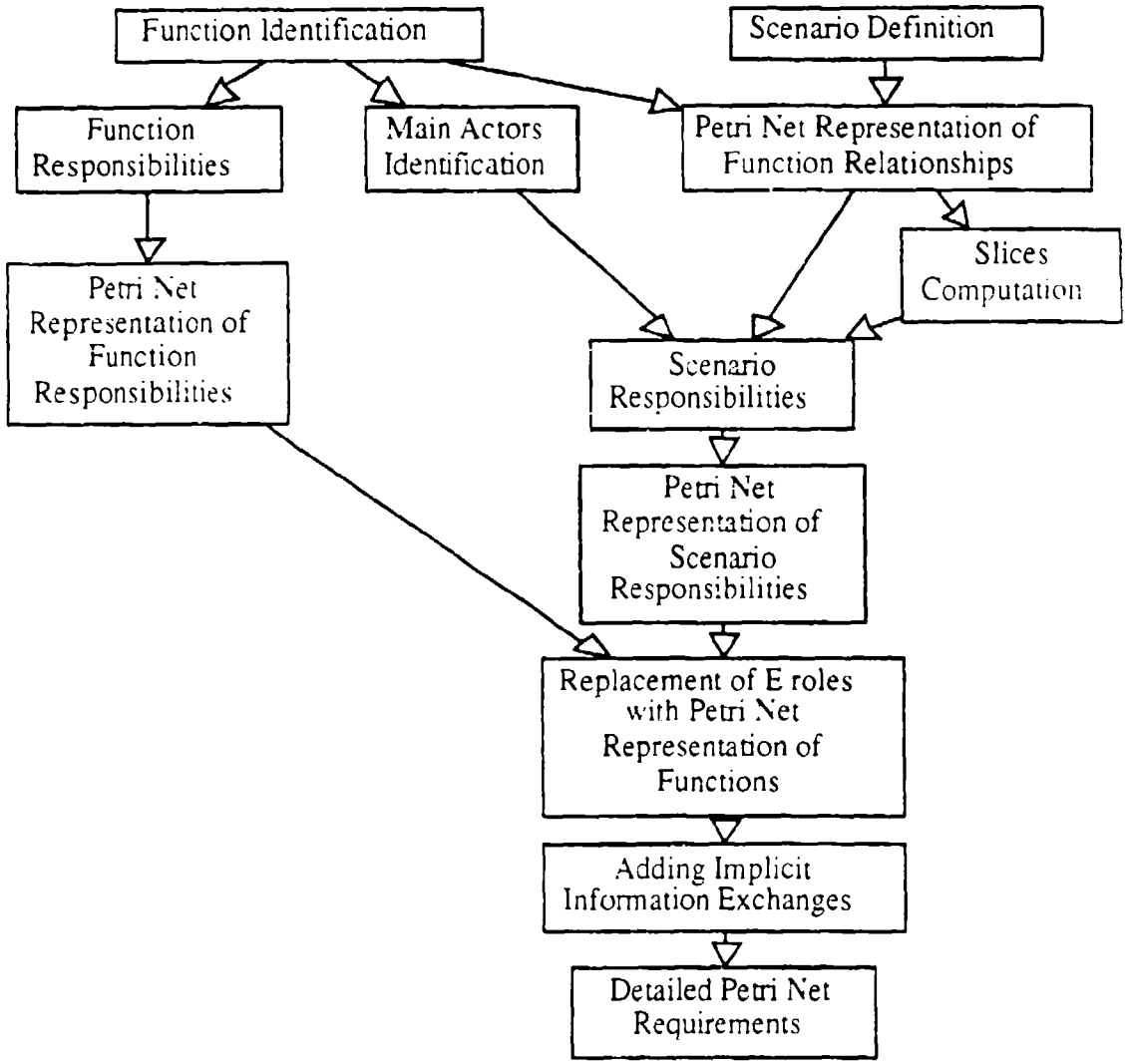


Figure 4.4 Procedures to Model the Detailed Requirements of a System

The definition of a scenario, that is a mission to be carried out, leads to the specification of the relationships and sequences of system functions. For the fulfillment of a mission, one can identify the system functions which can be executed concurrently as well as the functions which will have to be executed first to trigger the execution of a sequence of functions. These

interrelationships among functions vary from one scenario to another. Petri Nets are used to represent the sequencing and concurrency of functions so that the global requirements of a system can be derived. The procedure for determining the detailed requirements starts with the definition of the responsibilities for the chosen scenario. To list the functions on the Functions axis, the slices (Hillion and Levis,1987) of the Petri Nets representing the global requirements are computed. These slices represent the functions which can be executed concurrently. The functions are listed on this axis in the order of appearance in the slices list. Then, for each function, the actor which triggers the execution and gets the final report is identified. This actor is designated as the main one responsible for the execution of this functions. Once the main actors are listed on the Actor axis, the responsibility plane for the scenario can be constructed. For each function:

- A role E is placed on the cell defined by the function and by the main actor.
- Roles W are placed on the cells defined by the functions and by the main actors who are responsible for the execution of the subsequent functions as determined by the Petri Nets of the global requirements.

From the information in the scenario responsibilities plane, the equivalent Petri Net can be constructed following the same procedure that was used for the functions. The next step is to replace each transition representing a function with the equivalent representation of the responsibilities of this functions. By adding the implicit exchanges among actions for each actor, the Petri Net of the detailed requirements is constructed.

4.3 APPLICATION OF THE METHODOLOGY TO AN EXAMPLE

Let us consider an example where there are three functions: f1, f2 and f3, and three actors (A1, A2 and A3). The responsibilities for each function are defined as shown on Table 4.1.

Table 4.1 Responsibility Analysis Planes for f1, f2 and f3

f1	A1	A2	A3
s1	E	A	W
s2	C	E	A
s3		C	E
s4	C	E	W

f2	A2
s1	E

f3	A1	A3
s1	A	E
s2	E	C

The derived Petri Nets are displayed on Figure 4.5

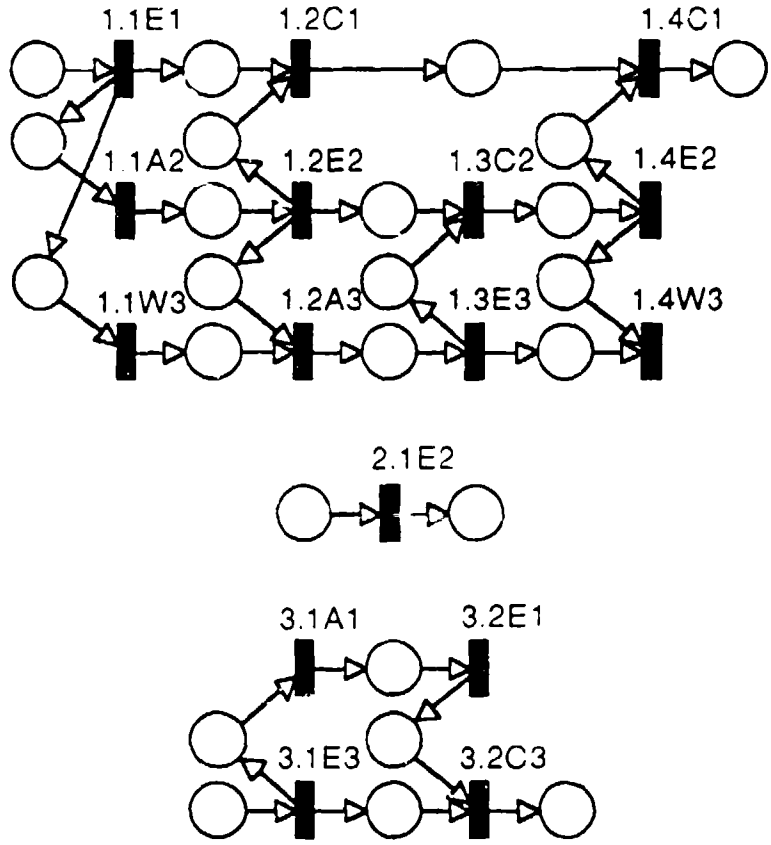


Figure 4.5 Petri Net Representation of the Responsibilities for f1, f2 and f3

The scenario specification has determined that f1 and f2 have to be executed before f3. The Petri Net of the Global requirements is shown on Figure 4.5. Its slices are:

Slice 1: f1, f2

Slice 2: f3

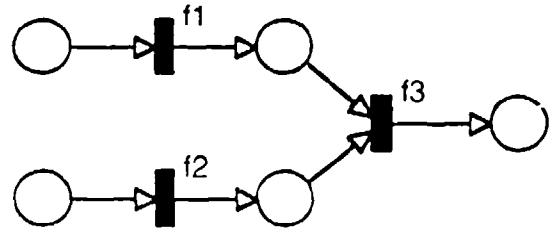


Figure 4.6 Global Requirements of the Example

The responsibilities specification of each function show that A1 is the main actor for f1, A2 for f2, and A3 for f3. The scenario responsibility plane is constructed by placing a role E in the cells (f1, A1), (f2, A2) and (f3,A3) and a role W in the cells (f1, A3) and (f2, A3), as shown on Table 4.2.

Table 4.2 Example of Scenario Requirements

	A1	A2	A3
f1	E		W
f2		E	W
f3			E

The derived Petri Net for the scenario responsibilities is shown on Figure 4.7

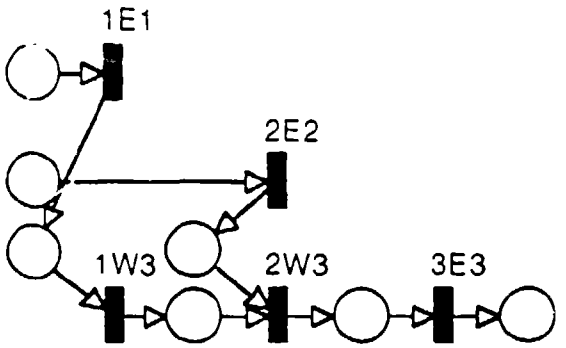


Figure 4.7 Scenario Responsibilities for the Example

By replacing each transition containing the letter E with the Petri Net representation of the responsibilities of the function this role E models and then by adding the implicit information exchanges between the functions performed by a single actor, the representation of the detailed requirements is obtained and shown on Figure 4.8.

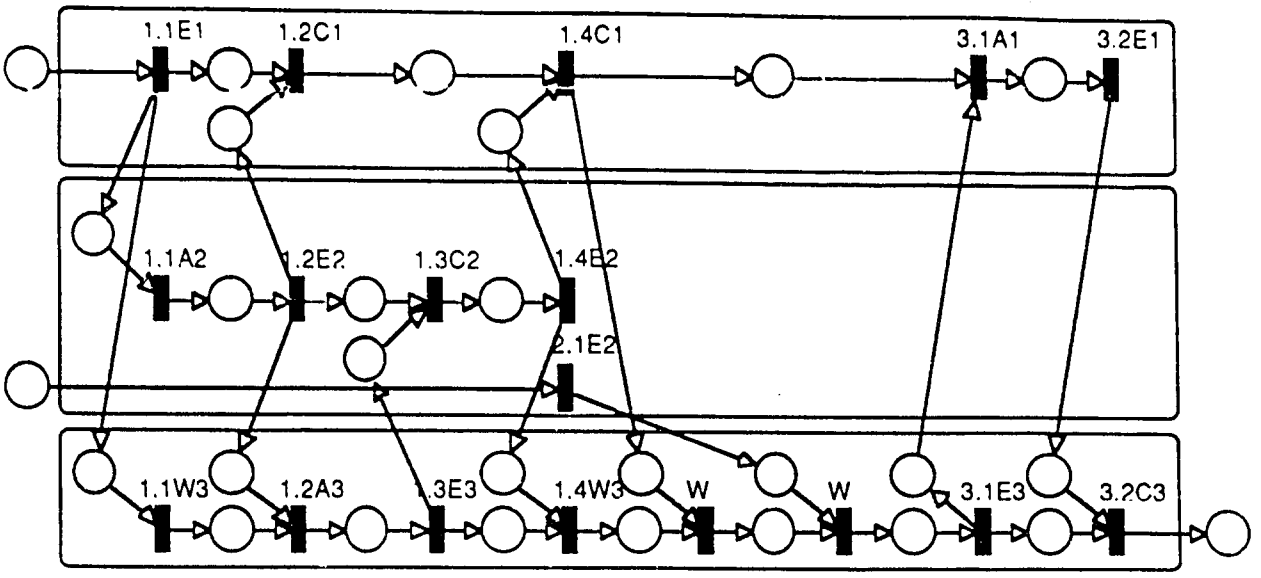


Figure 4.8 Detailed Requirements for the Example

4.4 MODELING REQUIREMENTS FOR DIFFERENT MODES OF OPERATION

For survivability reasons, the system must be able to switch to different modes of operation. In case of failures in a workstation or of partial destruction, the system has to be able to accomplish its mission. Therefore, the system designer must care for the duplication of software and interfaces on several workstations and for the the definition of the protocols for different modes of functioning. He has to define a nominal mode and back up modes which are in the Cube Tool framework reassignments of activities to different actors. These reassignments are done essentially at the function level and are done at the scenario level only to insure consistency throughout the functions (if a mode consists of ignoring one of the actor, the designer has to be sure that this actor is not involved in the execution of any function). For more clarity, the taking into account of different modes of operation will be described at the system level, since the procedure will be the same at the scenario level.

For each mode, there is a different allocation of activities to the actors for the definition of responsibilities. We consider that the decomposition of functions in subfunctions remains unchanged when switching from one mode to another. Therefore, a Petri Net representation of the responsibilities for a function can be derived for each mode of operation. The next step is to fold together these different nets to represent the responsibilities for all the modes of operation. This is done by using Colored Petri Nets.

The Colored Petri Nets that will be used have the following characteristics:

- The color of the tokens belongs to the set of the modes of operation $\{m_1, m_2, ..., m_n\}$
- The possible firing modes are the modes of operation of the system and correspond to the token-color as in the first example of section 2.2.3.
- The set of occurrence colors of each transition is constructed as follows: for each mode m_i, if the cell of the responsibility plane, defined by the subfunction and the actor, is non-null then m_i belongs to the set of occurrence color of the transition representing this activity, **regardless of the kind of processing this activity represents.** The label of the colored transition will indicate what type of processing it represents for the different modes of operation. For example, the label 1.2.E1.C2.3 indicates an activity of the subfunction 2 of the function 1 performed by actor 3 and which is of type E in mode 1 and of type C in mode 2.
- The set of tokens colors for each place and the matrices that annotate the arcs are determined by considering each set connector-place-connector and by looking if the transitions which are at the edge of this set are active during in the mode of operation.

Let us illustrate this with a simple example:

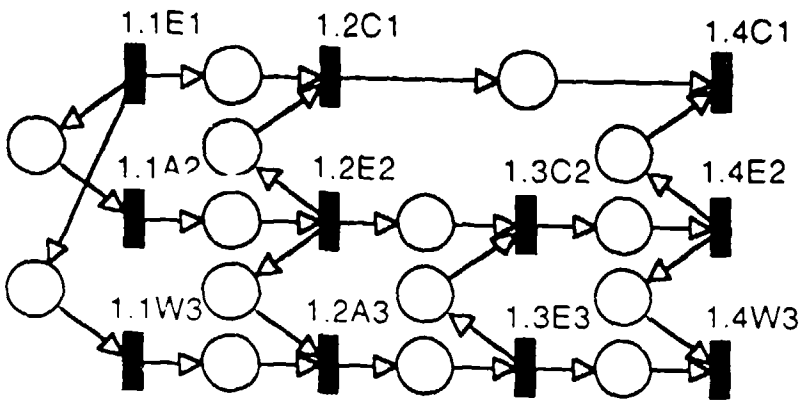
Let us consider a system with four actors A1, A2, A3 and A4, and let us assume that two modes of operations m1 and m2 have been defined. m1 is the nominal mode while m2 is the back up mode when the actor A2 is unable to accomplish its tasks because of failure or destruction. In this case A3 has to do the job of A2, and A4 accomplishes the job that A3 is doing in the nominal mode. The responsibility analysis planes for the function f1 in these two modes of operations are shown on Table 4.3.

Table 4.3 Responsibility Analysis Planes for f1 for the Two Modes of Operation

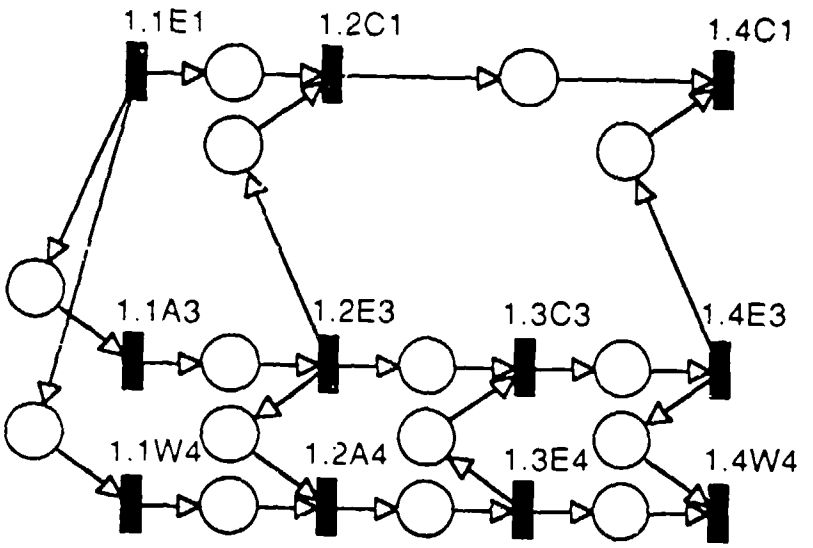
f1/m1	A1	A2	A3	A4
s1	E	A	W	
s2	C	E	A	
s3		C	E	
s4	C	E	W	

f1/m2	A1	A2	A3	A4
s1	E		A	W
s2	C		E	A
s3			C	E
s4	C		E	W

The derived Petri Net for each of these responsibility planes is shown on Figure 4.9.



Mode m1



Mode m2

Figure 4.9 Petri Nets of the Responsibilities for f1 for the Modes m1 and m2

The folding of the two nets depicted on Figure 4.9 leads to the Colored Petri Net shown on Figure 4.10.

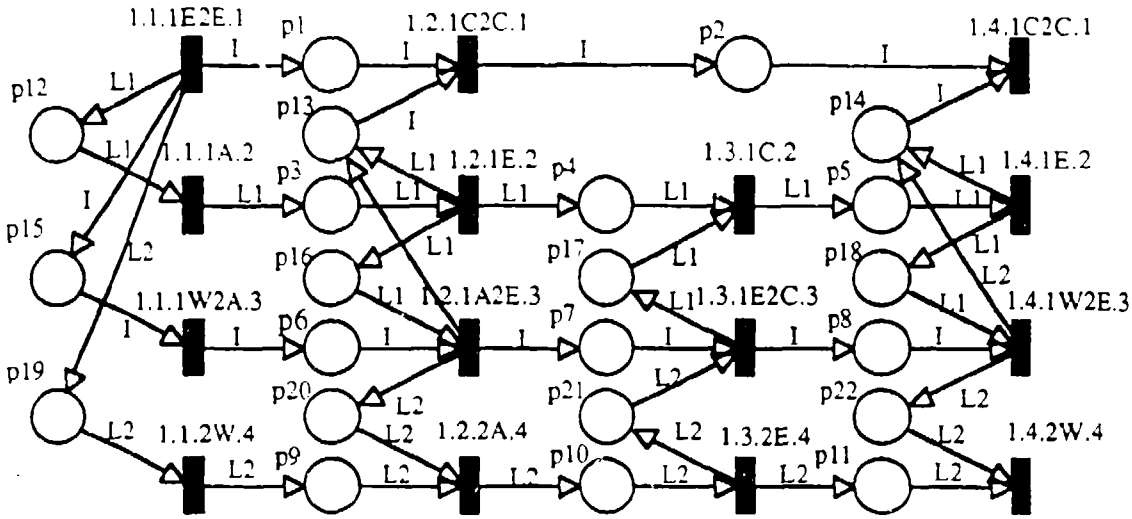


Figure 4.10 Colored Petri Net for f1 for the Modes m1 and m2

The Colored Petri Net of Figure 4.10 has the following characteristics:

The sets of occurrence-colors for each transition is:

C(1.1.1E2E.1) = C(1.2.1C2C.1) = C(1.4.1C2C.1) = {m1, m2}

C(1.1.1A.2) = C(1.1.1E.2) = C(1.3.1C.2) = C(1.4.1E.2) = {m1}

C(1.1.1W2A.3) = C(1.2.1A2E.3) = C(1.3.1E2C.3) = C(1.4.1W2E.3) = {m1, m2}

C(1.1.2W.4) = C(1.1.2A.4) = C(1.3.2E.4) = C(1.4.2W.4) = {m2}

The set of token-colors for each place is:

C(p1) = C(p2) = C(p6) = C(p7) = C(p8) = C(p13) = C(p14) = C(p15) = {m1, m2}

C(p3) = C(p4) = C(p5) = C(p7) = C(p12) = C(p16) = C(p17) = C(p18) = {m1}

C(p9) = C(p10) = C(p11) = C(p19) = C(p20) = C(p21) = C(p22) = {m2}

The three possible matrices that annotate the arcs are:

$$I = \begin{array}{c} \begin{matrix} m1 & m2 \end{matrix} \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{matrix} m1 \\ m2 \end{matrix} \end{array} \qquad L1 = \begin{array}{c} \begin{matrix} m1 & m2 \end{matrix} \\ \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{matrix} m1 \\ m2 \end{matrix} \end{array} \qquad L2 = \begin{array}{c} \begin{matrix} m1 & m2 \end{matrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{matrix} m1 \\ m2 \end{matrix} \end{array}$$

The next chapter describes the application of the methodology to a real example of C3I system.

CHAPTER 5

EXAMPLE: AN AIR INTERDICTION MISSION SYSTEM

5.0 INTRODUCTION

The system used to illustrate the methodology is a fictitious one called MESACC, which stands for Modular, Endurable, Survivable, Austere, Command Center and which has been studied by Valraud, Perdu and Levis and described in Valraud and Levis (1989).

The objective of an air interdiction mission system is to plan operations against the enemy's military potential before it can be effectively used against friendly forces. These operations restrict the combat capability of the enemy by:

- delaying, disrupting, or destroying his lines of communications
- destroying enemy supplies
- attacking fixed, moving and movable point and area targets
- destroying unengaged or uncommitted enemy attack formations before they can be brought into the battle.

The result of these operations is to disrupt enemy plans and time schedules. The integration of air interdiction operations with the fire and maneuver plans of surface forces is not required. However, these offensive air operations are planned and conducted as part of the unified effort of all friendly forces. Therefore, air interdiction demands precise coordination and timing.

5.1 FUNCTION IDENTIFICATION

The identification of the functions of the system requires the examination of the context and the environment in which the system operates. The context consists of the geographical characteristics of the battle area. It is assumed that the system is operating in Europe, and more specifically in the central region. The environment consists of the friendly forces, their assets, strength, current plans and orders, the enemy forces, their assets, strength, current plans and orders. Also, the current weather is part of the environment as it is a particularly important factor in air interdiction mission planning. The functions needed to plan an Air Interdiction Mission are listed in Table 5.1. These functions are described further in this chapter.

For each function, subfunctions have been defined. Some of them are used in different functions and for the purpose of clarity, a unique identification number has been assigned to each one which is used consistently in the figures and tables. The list is given in Table 5.2.

Table 5.1 System Functions of MESACC

Function #	Description
1	Weather Projection
2	Format Messages/Information Fusion/Update Database
3	Status of Allied Forces
4	Strike Assesment
5	Threat Assesment
6	Current Intelligence
7	Target Development/Prioritization
8	Aimpoint Construction/Weaponeering
9	Penetration/Attrition Analysis
10	Mission Planning
11	Weapon System Availability

Table 5.2 Subfunctions used in MESACC

Subfunction #	Description	Subfunction #	Description
1	Acknowledge	27	Request Strike Assessment Data
2	Request Weather Data	28	Get Strike Assessment Data
3	Get Weather Data	29	Assess Strike
4	Deduce Weather Projection	30	Update Strike Assessment DataBase
5	Update Weather Data Base	31	Request Strike Report
6	Request Weather Projection	32	Generate Strike Report
7	Get Weather Projection	33	Request Threat Assessment Data
8	Request Allied Data	34	Get Threat Assessment Data
9	Get Allied Forces Data	35	Update Threat Assessment DataBase
10	Fuse Allied Forces Information	36	Modify Threat Assessment
11	Update Allied Forces DataBase	37	Rank Threats
12	Determine Allied Status	38	Request Targets List
13	Request Allied Report	39	Get Targets list
14	Get Allied Report	40	Request Available Weapons
15	Request Enemy Forces Data	41	Get Available weapons
16	Get Enemy Forces Data	42	Request Weapon Data
17	Fuse Enemy Forces Information	43	Get Weapon Data
18	Update Enemy Forces DataBase	44	Generate new Weapon status
19	Request Enemy Report	45	Generate List of Available Weapons
20	Generate Enemy Report	46	Request Current Intelligence
21	Request Battlefield Data	47	Get Current Intelligence
22	Get Battlefield Data	48	Aimpoint Construction Weaponeering
23	Fuse Battlefield Information	49	Perform Analysis
24	Update Battlefield DataBase	50	Request Penetration Analysis
25	Request Battlefield Report	51	Get Penetration Analysis
26	Generate Battlefield Report	52	Plan Mission

5.2 ACTOR IDENTIFICATION

For the identification of the actors, eleven workstations (WS) are considered, located in two shelters, and seven databases. In addition, the intelligence center is considered as an actor providing the latest information about the situation. In this example, databases are considered to be actors because they are distributed and exchanges have to take place on the network to access them. These seven databases are:

- DB-wt: contains weather forecasts that come from the weather reports and which are produced by f1.
- DB-wp: contains data about weapons availability, given the status of the allied equipment, and the weather forecasts.
- DB-en: contains data about the enemy .
- DB-ba: contains data about the situation on the battle field.
- DB-st: contains data about strike assessment, i.e., the result of f4.
- DB-th: contains data about threat assessment, i.e., the results of f5.
- DB-al: contains data about the allied forces.

There are therefore nineteen actors listed in Table 5.3 with their corresponding notation.

Table 5.3 The Nineteen Actors of MESACC

Actor #	Description	Notation
1	Workstation 1	WS1
2	Workstation 2	WS2
3	Workstation 3	WS3
4	Workstation 4	WS4
5	Workstation 5	WS5
6	Workstation 6	WS6
7	Workstation 7	WS7
8	Workstation 8	WS8
9	Workstation 9	WS9
10	Workstation 10	WS10
11	Workstation 11	WS11
12	Intelligence Center	INC
13	Data Base Weather	DB-wt
14	Data Base Weapons	DB-wp
15	Data Base Enemy Forces	DB-en
16	Data Base Battlefield	DB-ba
17	Data Base Strike Assesment	DB-st
18	Data Base Threats	DB-th
19	Data Base Allied Forces	DB-al

5.3 RESPONSIBILITIES SPECIFICATION FOR EACH FUNCTION

The responsibilities specification for each function is made by allocating roles to different the different actors involved in its execution.

5.3.1 Function 1: Weather Projection

This function forecasts the weather from the current weather reports.

When WS1 receives a weather report, he accesses the weather database (subfunctions "Request Weather Data" and "Get Weather Data") to get the relevant data to make the new weather projection (subfunction "Deduce Weather Projection"). He updates the database with this new weather projection (subfunctions "Update Weather Database" and 'Acknowledge"). Table 5.4 shows the responsibility plane for the function f1. The Petri Net is shown on Figure 5.1.

Table 5.4 Responsibilities for Function f1

f1	Weather Projection	Actor#	1	13
Subf#	Subfunction	Actor	WS1	Db-wt
2	Request Weather Data		E	A
3	Get Weather Data		C	E
4	Deduce Weather Projection		E	
5	Update Weather Data Base		E	A
1	Acknowledge		C	E

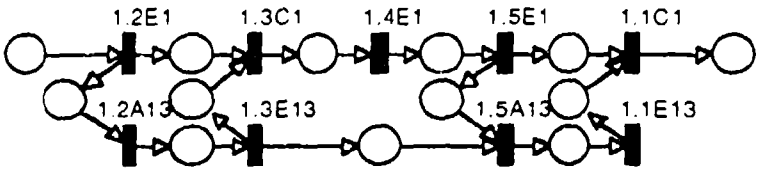


Figure 5.1 Petri Net for Function f1

5.3.2 Function 2: Format Messages/Fusion of Information/Update Databases

This function transforms the format of the various data inputs into a common format. The function also performs decoding. Then, this function updates the current information as new messages come in the system. For example, if the database contains the position of a particular enemy battalion, and later an intelligence report confirms that the battalion has moved to another

position, then the function is used to update the position of that battalion, its strength, and current plans.

WS11 takes care of reports about allied forces, while WS7 deals with reports about the enemy forces and WS8 with the battlefield situation. They work concurrently. When they receive reports, they have to access the database to get the relevant data about the situation (subfunctions "Request Allied Forces Data" and "Get Allied Forces Data" for WS11, "Request Enemy Forces Data" and "Get Enemy Forces Data" for WS7, "Request Battlefield Data" and "Get Battlefield Data" for WS8) to infer the new situation with the new information received (subfunctions "Fuse Allied Forces Information" for WS11, "Fuse Enemy Forces Information" for WS7, "Fuse Battlefield Information" for WS8). This situation is then stored in the database (subfunctions "Update Allied Forces Database" and "Acknowledge" for WS11, "Update Enemy Forces Database" and "Acknowledge" for WS7, "Update Battlefield Database" and "Acknowledge" for WS8). The Responsibility plane for the function f2 is shown on Table 5.5 and the corresponding Petri Net is displayed on Figure 5.2.

Table 5.5 Responsibilities for Function f2

f2	Form. Mess./IF/Update DB	Actor#	7	15	8	16	11	19
Subf#	Subfunction	Actor	WS7	DB-en	WS8	DB-ba	WS11	DB-al
8	Request Allied Data						E	A
9	Get Allied Forces Data						C	E
10	Fuse Allied Forces Information						E	
11	Update Allied Forces DataBase						E	A
1	Acknowledge						C	E
15	Request Enemy Forces Data		E	A				
16	Get Enemy Forces Data		C	E				
17	Fuse Enemy Forces Information		E					
18	Update Enemy Forces DataBase		E	A				
1	Acknowledge		C	E				
21	Request Battlefield Data				E	A		
22	Get Battlefield Data				C	E		
23	Fuse Battlefield Information				E			
24	Update Battlefield DataBase				E	A		
1	Acknowledge				C	E		

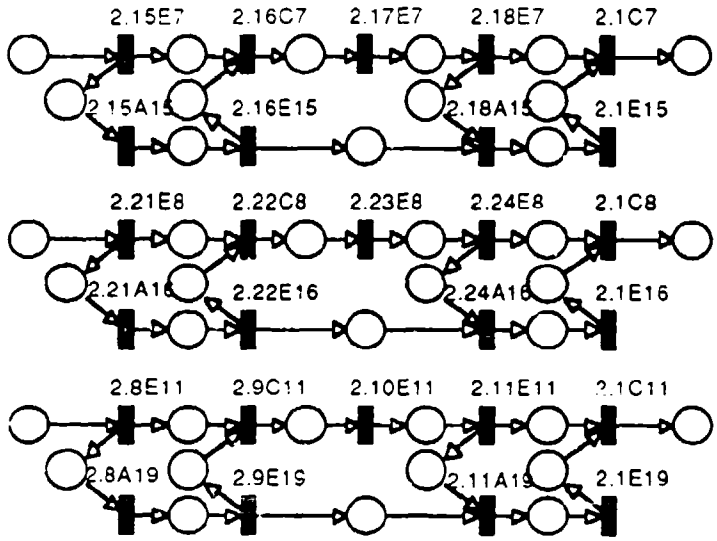


Figure 5.2 Petri Net for Function f2

5.3.3 Function f3: Status of Allied Forces.

This function is used to assess the current state of the allied forces, number of troops, equipment, and of available aircraft for missions.

To determine the status of allied forces, Workstation WS7 needs to get information from the battlefield and to deduce from the last state of the allied forces the new status. Therefore, WS8 is queried to obtain a battlefield report (subfunction "Request Battlefield Report"). To do this, WS8 has to access the database Battlefield (subfunctions "Request Battlefield Data" and "Get Battlefield Data") to make the report that it sends to WS7 (subfunction "Generate Battlefield Report"). According to the data that WS7 has just received, WS7 accesses the allied forces data base (subfunction "Request Allied Data" and "Get Allied Data") to determine the new status of the allied forces (subfunction "Determine Allied Status"). Once this is performed, the allied forces database has to be updated (subfunctions "Update Allied Forces" and "Acknowledge"). The responsibility analysis plane is shown on Table 5.6 and the Petri Net deduced from this responsibility analysis plane is displayed on Figure 5.3.

Table 5.6 Responsibilities for Function f3

f3	Status of Allied Forces	Actor#	7	8	16	19
Sub#	Subfunction	Actor	WS7	WS8	DB-ba	DB-al
25	Request Battlefield Report		E	A		
21	Request Battlefield Data			E	A	
22	Get Battlefield Data			C	E	
26	Generate Battlefield Report		C	E		
8	Request Allied Data		E			A
9	Get Allied Forces Data		C			E
12	Determine Allied Status		E			
11	Update Allied Forces DataBase		E			A
1	Acknowledge		C			E

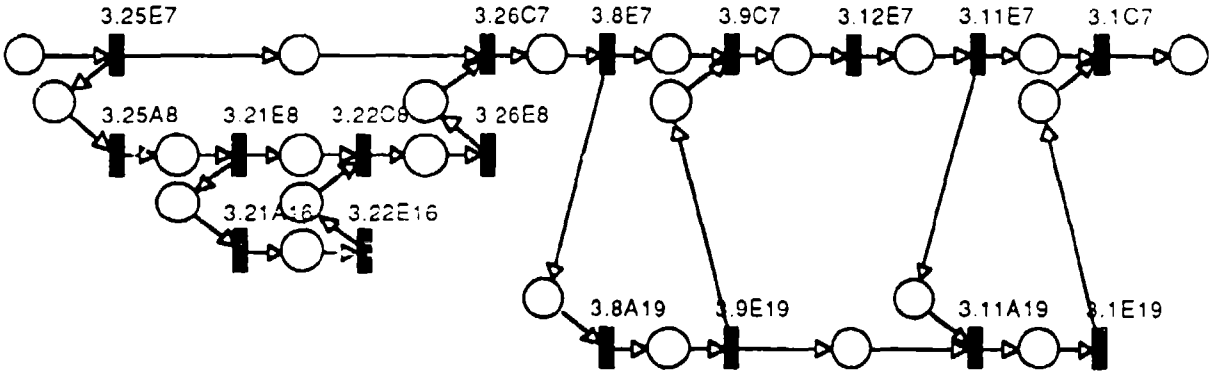


Figure 5.3 Petri Net for Function f3

5.3.4 Function 4: Strike Assessment

This function updates the situation on the battlefield, as a result of previous air interdiction missions.

WS8 accesses the Strike Assessment Database to get the last assessment made (subfunctions "Request Strike Assessment Data" and "Get Strike Assessment Data"). He then queries WS7 to get a report on the enemy forces (subfunction "Request Enemy Report"). WS7 accesses the Enemy Forces Database (subfunctions "Request Enemy Forces Data" and "Get Enemy Forces Data") to make the report he sends to WS8 (subfunction "Generate Enemy Report"). Then, WS8 has to access the Threat Assessment Database (subfunctions "Request Threat Assessment Data" and "Get Threat Assessment Data") to check what threats have been destroyed or what new threat has appeared. From all the information he can then assess the result of previous air interdiction missions and what remains to be done (subfunction "Assess Strike"). WS8 finally stores these

information in the Strike Assessment Database (subfunctions "Update Strike Assessment Database"). The responsibility analysis plane is shown on Table 5.7 and the Petri Net derived from this responsibility analysis plane is displayed on Figure 5.4.

Table 5.7 Responsibilities for Function f4

f4	Strike Assesment	Actor#	7	15	8	17	18
Subf#	Subfunction	Actor	WS7	DB-en	WS8	DB-st	DB-th
27	Request Strike Assessment Data				E	A	
28	Get Strike Assessment Data				C	E	
19	Request Enemy Report		A		E		
15	Request Enemy Forces Data		E	A			
16	Get Enemy Forces Data		C	E			
20	Generate Enemy Report		E		C		
33	Request Threat Assessment Data				E		A
34	Get Threat Assessment Data				C		E
29	Assess Strike				E		
30	Update Strike Assessment DataBase				E	A	
1	Acknowledge				C	E	

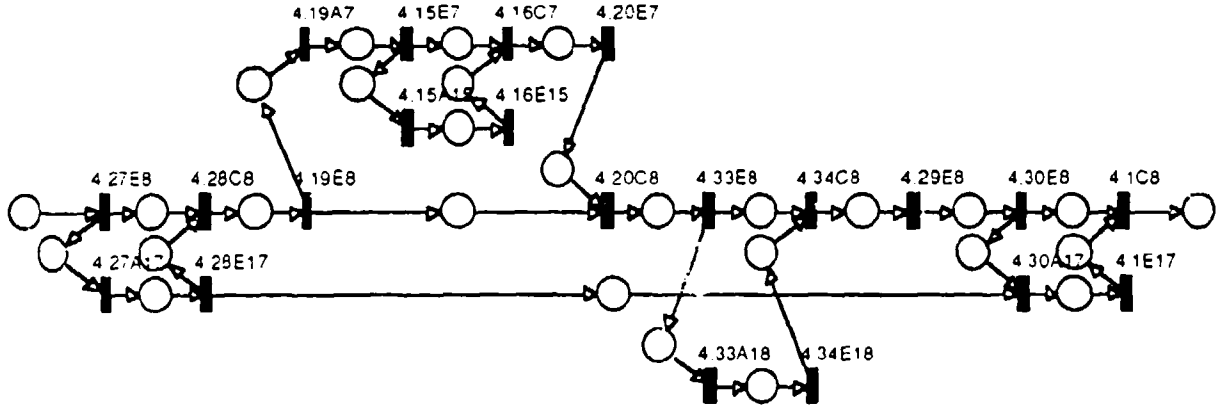


Figure 5.4 Petri Net for Function f4

5.3.5 Function 5: Threat Assessment

This function evaluates the threat of the enemy forces in the different sectors of the battlefield.

To make the threat assessment, WS9 needs to get the last threat assessment from the Threat Assessment Database (subfunctions "Request Threat Assessment Data" and "Get Threat Assessment Data"). He asks WS7 to report on the enemy forces (subfunctions "Request Enemy

Report", "Request Enemy Forces Data", "Get Enemy Forces Data" and "Generate Enemy Report") and WS8 to report on the battlefield situation (subfunctions "Request Battlefield Report", "Request Battlefield Data", "Get Battlefield Data" and "Generate Battlefield Report"). From these reports, he can build a list of the threats that he stores in the Threat Assessment Database (subfunctions "Update Threat Assessment Database", and "Acknowledge"). The responsibility analysis plane is shown on Table 5.8 and the derived Petri Net is displayed on Figure 5.5.

Table 5.8 Responsibilities for Function f5

f5	Threat Assesment	Actor#	7	15	8	16	9	18
Subf#	Subfunction	Actor	WS7	DB-en	WS8	DB-ba	A9	DB-th
33	Request Threat Assessment Data		W		W		E	A
34	Get Threat Assessment Data						C	E
19	Request Enemy Report		A				E	
15	Request Enemy Forces Data		E	A			C	
16	Get Enemy Forces Data		C	E				
20	Generate Enemy Report		E				C	
25	Request Battlefield Report		E		A			
21	Request Battlefield Data				E	A		
22	Get Battlefield Data				C	E		
26	Generate Battlefield Report				E		C	
35	Update Threat Assessment DataBase						E	A
1	Acknowledge						C	E

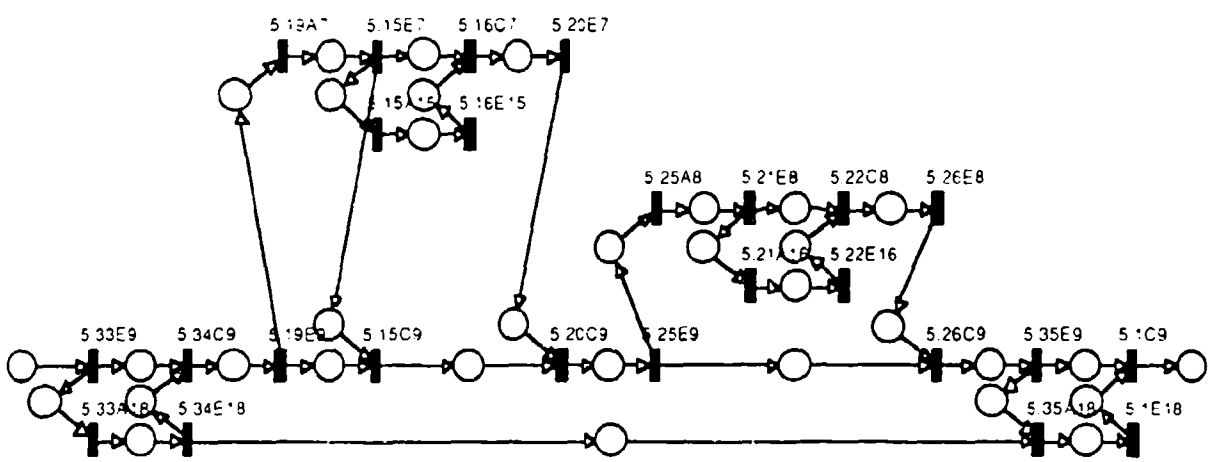


Figure 5.5 Petri Net for Function f5

5.3.6 Function 6: Current Intelligence

Under certain circumstances, reports from intelligence may be requested when the uncertainty about some parameters of the problem is deemed too high.

If the information contained in the databases is not sufficient to make the threat assessment, WS9 can ask the Intelligence Center (INC) (subfunctions "Request Current Intelligence" and "Get Current Intelligence"), if it has any current intelligence on the battlefield situation. With this new information, WS9 can then modify the threat assessment (subfunction "Modify Threat Assessment"). The responsibility analysis plane is shown on Table 5.9 and the Petri Net deduced from this responsibility analysis plane is displayed on Figure 5.6.

Table 5.9 Responsibilities for Function f6

f6	Current Intelligence	Actor#	9	12
Subf#	Subfunction	Actor	WS9	INC
46	Request Current Intelligence		E	A
47	Get Current Intelligence		C	E
36	Modify Threat Assessment		E	

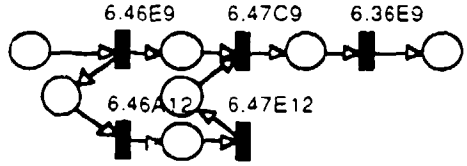


Figure 5.6 Petri Net for Function f6

5.3.7 Function 7: Target Prioritization/Target Development

This function is needed to prioritize the most important objective to be destroyed, given the situation. The resources that can be used for the next mission may be scarce so that it may not be possible to allocate assets to all objectives.

To do this, WS2 needs to get the threat list from the Threat Assessment Database (subfunctions "Request Threat Assessment Data" and "Get Threat Assessment Data"). WS2 needs also a report on the situation on the battlefield (subfunction "Request Battlefield Report") that he asks WS6, who has to access the Battlefield Database (subfunctions "Request Battlefield Data" and "Get Battlefield Data"), to generate this report (subfunction "Generate Battlefield Report"). Once WS2 receives this report, he can rank the threats according to the current

situation (subfunction "Rank Threats"). The responsibility analysis plane is shown on Table 5.10 and the Petri Net deduced from this responsibility analysis plane is displayed on Figure 5.7.

Table 5.10 Responsibilities for Function f7

f7	Target Develop./Priorit.	Actor#	2	6	16	18
Subf#	Subfunction	Actor	WS2	WS6	DB-ba	DB-th
33	Request Threat Assessment Data		E			A
34	Get Threat Assessment Data		C			E
25	Request Battlefield Report		E	A		
21	Request Battlefield Data		C	E	A	
22	Get Battlefield Data			C	E	
26	Generate Battlefield Report		C	E		
37	Rank Threats		E			

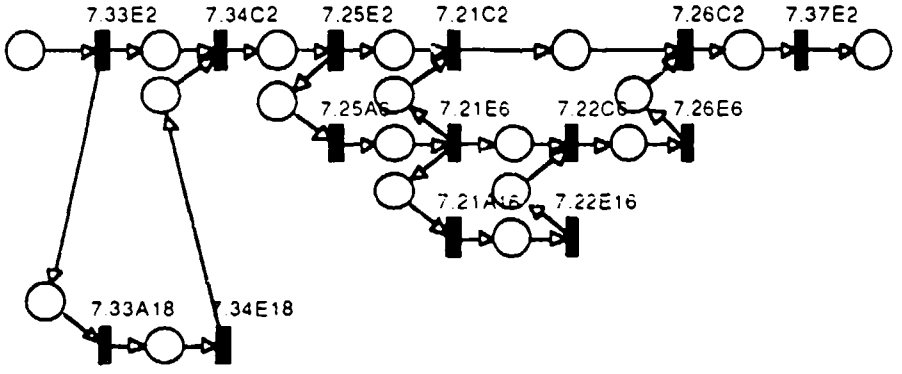


Figure 5.7 Petri Net for Function f7

5.3.8 Function 8: Aimpoint Construction/ Weaponeering

This function provides the coordinates of the target, and allocates certain classes of friendly assets according to the objective and its intrinsic characteristics. WS4 is in charge of executing this. The responsibility analysis plane is shown on Table 5.11 and the Petri Net deduced from this responsibility analysis plane is displayed on Figure 5.8.

Table 5.11 Responsibilities for Function f8

f8	Aimpoint Construct./Weapon	Actor#	4
Subf#	Subfunction	Actor	WS4
48	Aimpoint Construction Weaponeering		E

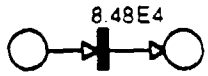


Figure 5.8 Petri Net for Function f8

5.3.9 Function 9: Penetration/Attrition Analysis

This function forecasts the degree of redundancy that is adequate for each objective. Different platforms may be assigned the same objective to protect friendly assets. Therefore, redundancy insures a greater degree of certainty over the outcome of the mission.

The penetration/attrition analysis is performed by WS3. To do this, he needs a report on the position of the enemy that he obtains from WS10 (subfunctions "Request Enemy report", "Request Enemy Forces Data", "Get Enemy Forces Data" and "Generate Enemy Report"). The Strike Assessment report that WS3 asks from WS6, allows him to know to what extent the defenses of the enemy have been reduced as a result of previous missions (subfunctions "Request Strike report", "Request Strike Assessment Data", "Get Strike Assessment Data" and "Generate Strike Report"). Finally, to avoid to harm to allied forces, he needs to know their position (subfunctions "Request Allied Forces Data" and "Get Allied Forces Data") before performing the analysis (subfunction "Perform Analysis"). The responsibility analysis plane is shown on Table 5.12 and the Petri Net deduced from this responsibility analysis plane is displayed on Figure 5.9.

Table 5.12 Responsibilities for Function f9

f9	Penetration/Attrition Anal.	Actor#	3	6	15	17	10	19
Subf#	Subfunction	Actor	WS3	WS6	DB-en	DB-st	WS10	DB-al
19	Request Enemy Report		E				A	
15	Request Enemy Forces Data				A		E	
16	Get Enemy Forces Data				E		C	
20	Generate Enemy Report		C				E	
31	Request Strike Report		E	A				
27	Request Strike Assessment Data		C	E		A		
28	Get Strike Assessment Data			C		E		
32	Generate Strike Report		C	E				
8	Request Allied Data		E					A
9	Get Allied Forces Data		C					E
49	Perform Analysis		E					

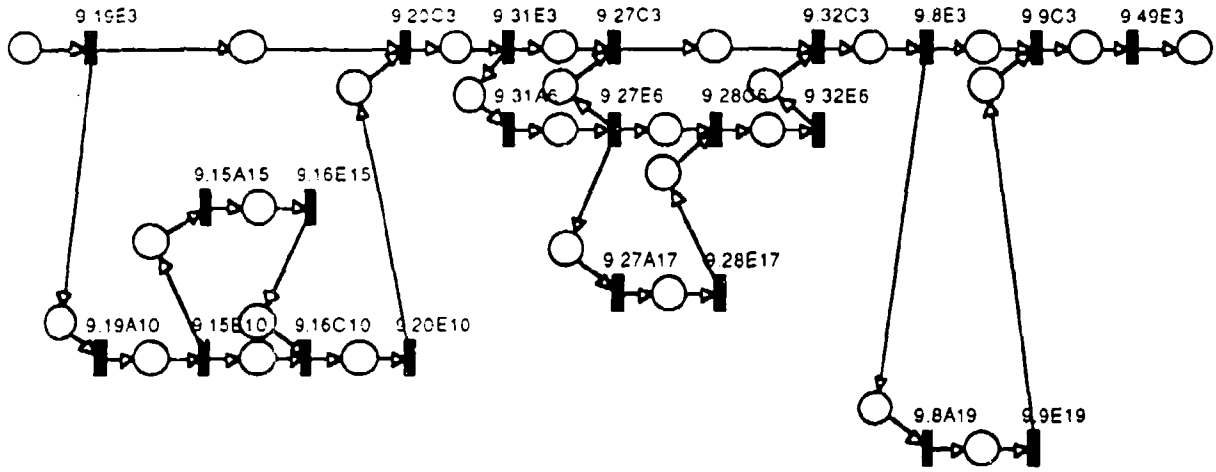


Figure 5.9 Petri Net for Function f9

5.3.10 Function 10: Mission Planning

This function delivers the final output of the system to the environment. It consists of a set of missions with the objectives, the type of aircraft to be used, the armament, the number of aircraft to be used for each objective, the route to be followed, and the time to perform the mission.

WS4 is in charge of planning the mission and generate the order. He asks WS2 for the targets list (subfunctions "Request Targets List" and "Get Targets List"). He needs to know from WS5 what weapons systems are available (subfunctions "Request Available Weapons" and "Get Available Weapons") and also the weather projection from WS1 (subfunctions "Request Weather Projection" and "Get Weather Projection") to determine what type of missions are possible. The penetration analysis given to him by WS3 (subfunctions "Request Penetration Analysis" and "Get Penetration Analysis") allows to select the targets and to plan the mission (subfunction "Plan Mission"). The responsibility analysis plane is shown on Table 5.13 and the Petri Net deduced from this responsibility analysis plane is displayed on Figure 5.10.

Table 5.13 Responsibilities for Function f10

f10	Mission Planning	Actor#	1	2	3	4	5
Subf#	Subfunction	Actor	WS1	WS2	WS3	WS4	WS5
38	Request Targets List			A		E	
39	Get Targets list			E		C	
40	Request Available Weapons					E	A
41	Get Available weapons					C	E
6	Request Weather Projection		A			E	
7	Get Weather Projection		E			C	
50	Request Penetration Analysis				A	E	
51	Get Penetration Analysis				E	C	
52	Plan Mission					E	

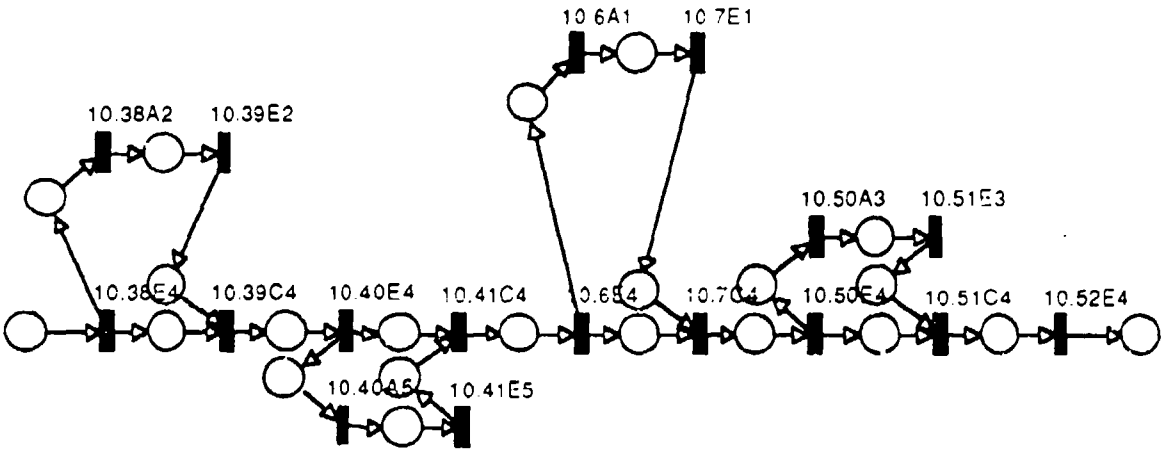


Figure 5.10 Petri Net for Function f10

5.3.11 Function 11: Weapon System Availability

This function describes what weapons are available for the mission at the time it is planned. This function tells what is available according to the weather forecasts (some aircraft cannot fly under certain circumstances), and the status of the allied forces (losses, use of reserves).

WS5 gets from the Weapon Database the last weapon status (subfunctions "Request Weapon Data" and "Get Weapon Data"). The report on the allied status that he asks from WS10 (subfunctions "Request Allied report", "Request Allied Data" Get Allied Data" and "Generate Allied Report") allows him to check what has been damaged and to generate a new weapon status that he stores in the Weapon Database (subfunctions "Generate New Weapon Status" and "Acknowledge"). The weather data he gets from the Weather Database (subfunctions "Request Weather Data" and "Get Weather Data") allows him to generate a list of available weapons that

can be used in the current weather situation (subfunction "Generate List of Available Weapons"). The responsibility analysis plane is shown on Table 5.14 and the Petri Net deduced from this responsibility analysis plane is displayed on Figure 5.11.

Table 5.14 Responsibilities for Function f11

f11	Weapon System Availability	Actor#	13	5	14	10	19
Subf#	Subfunction	Actor	DB-wt	WS5	DB-wp	WS10	DB-al
42	Request Weapon Data			E	A		
43	Get Weapon Data			C	E		
13	Request Allied Report			E		A	
8	Request Allied Data			C		E	A
9	Get Allied Forces Data					C	E
14	Get Allied Report			C		E	
44	Generate new Weapon status			E	A		
1	Acknowledge			C	E		
2	Request Weather Data		A	E			
3	Get Weather Data		E	C			
45	Generate List of Available Weapons			E			

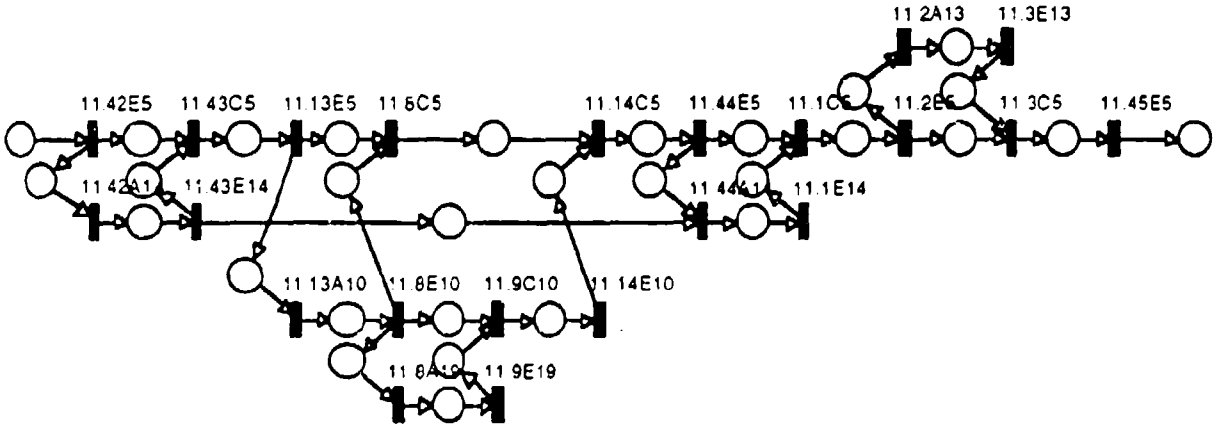


Figure 5.11 Petri Net for Function f11

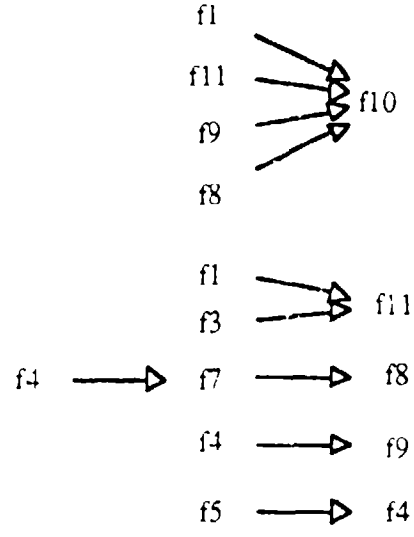
5.4 GENERATING THE DETAILED REQUIREMENTS

5.4.1 Global Requirements

A specific scenario is considered next. It is assumed that the hostilities started two days ago. Although the enemy has gained ground on the battlefield, the friendly forces resist the pressure,

and major assets in reserve have not been committed on either side. The conflict is a conventional one. The friendly forces and the enemy forces have both fairly accurate information about the situation on the other side. Each side knows what the resources are on the opposing side, as well as the location of these assets, although some uncertainty remains. In certain areas, the battle line is difficult to assess. Therefore, there is a need to use MESACC to plan long distance bombing from high altitude. In this scenario, some functions are not necessary, such as the "construct database" function. Since the conflict started two days ago, the database already exists. All the other functions described earlier are in use.

The various data inputs from the sensors are weather reports, reports on friendly and enemy forces (strength, position, status), combat reports, request from the local Command Center for assistance, mission reports, and current and future operations plans. The output is unique and consists of air interdiction mission plans. The interrelationship between the various functions is as follows:



It should be understood that this description of the interrelationship between functions is purely functional. If a function is derived from another, it does not mean that the input of that function is sufficient. Indeed, data from the context may be necessary (terrain information for example). The global functional requirements of MESACC are described by the Petri Net of Figure 20. This Petri Net contains only one switch, s1, which represents the optional use of the "Current Intelligence" function, f6.

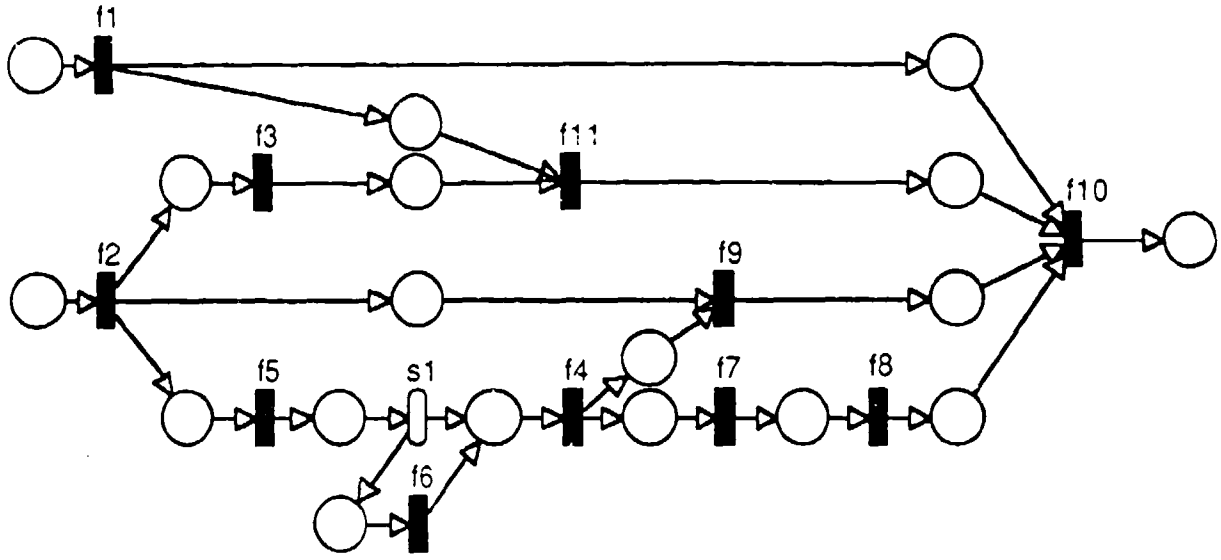


Figure 5.12 MESACC: The Global Functional Requirements

5.4.2 Detailed Requirements

The Petri Net obtained from the global functional requirements is used to determine the slices of the Net. The slices are:

Slice 1: f1, f2.

Slice 2: f3, f5.

Slice 3: f6.

Slice 4: f4,f11.

Slice 5: f7, f9.

Slice 6: f8.

Slice 7: f10.

To determine how data are transmitted among functions, Cube Tool has to be applied once more. The purpose is to define the global responsibility for the scenario. In the definition of the responsibilities for each function, only one actor triggers the execution and gets the final report. By looking at the global functional requirements, and at the different responsibility planes, one can identify where the output of each function has to be sent in order to generate the Scenario Responsibilities Plane shown on Table 5.15. The Petri Net of the scenario is deduced and shown on Figure 5.13.

Table 5.15 Scenario Responsibilities

	WS1	WS2	WS3	WS4	WS5	WS7	WS8	WS9	WS10
f1	E			W	W				
f2			W			W	W	W	E
f3					W	E			
f5								E	
f6							W	E	
f11				W	E				
f4		W	W				E		
f7		E		W					
f8				E					
f9			E	W					
f10				E					

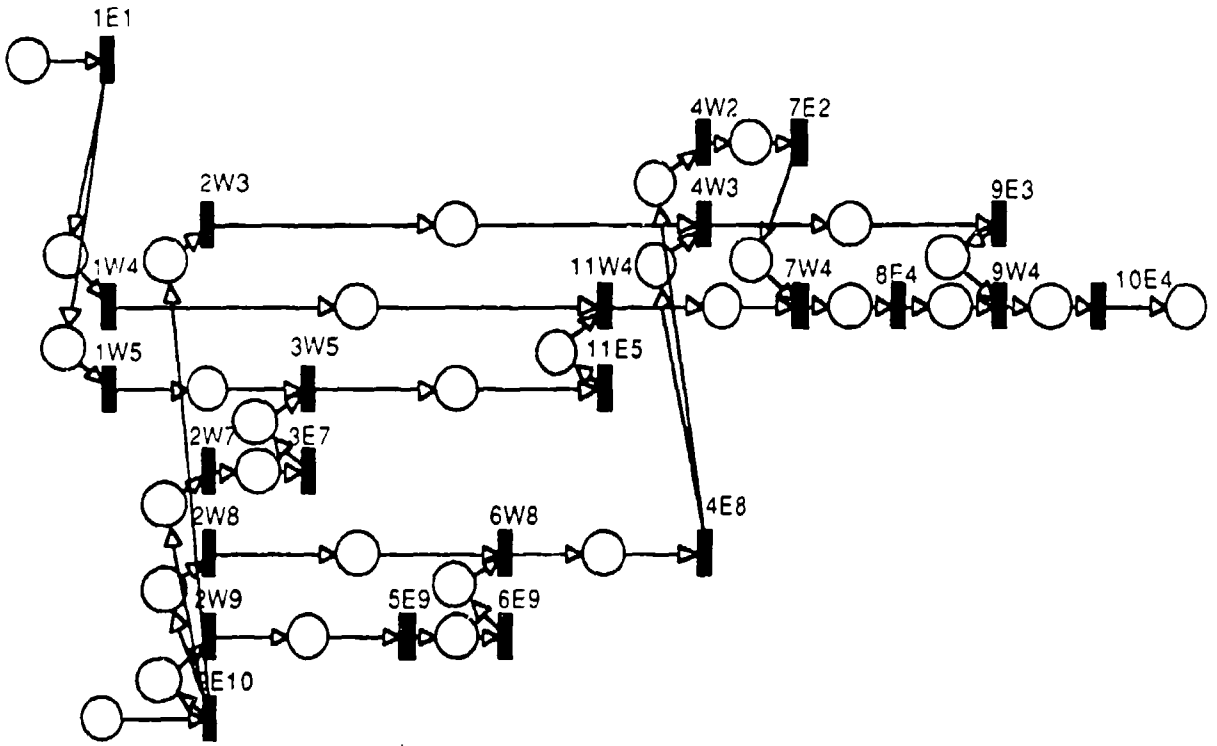


Figure 5.13 Petri Net of the Scenario

The representation of the detailed requirements is obtained (1) by replacing each transition containing the letter E with the Petri Net representation of the responsibilities of the functions this role E models and (2) by adding the implicit information exchanges between the functions performed by a single actor. Figure 5.14 displays the detailed requirements of MESACC.

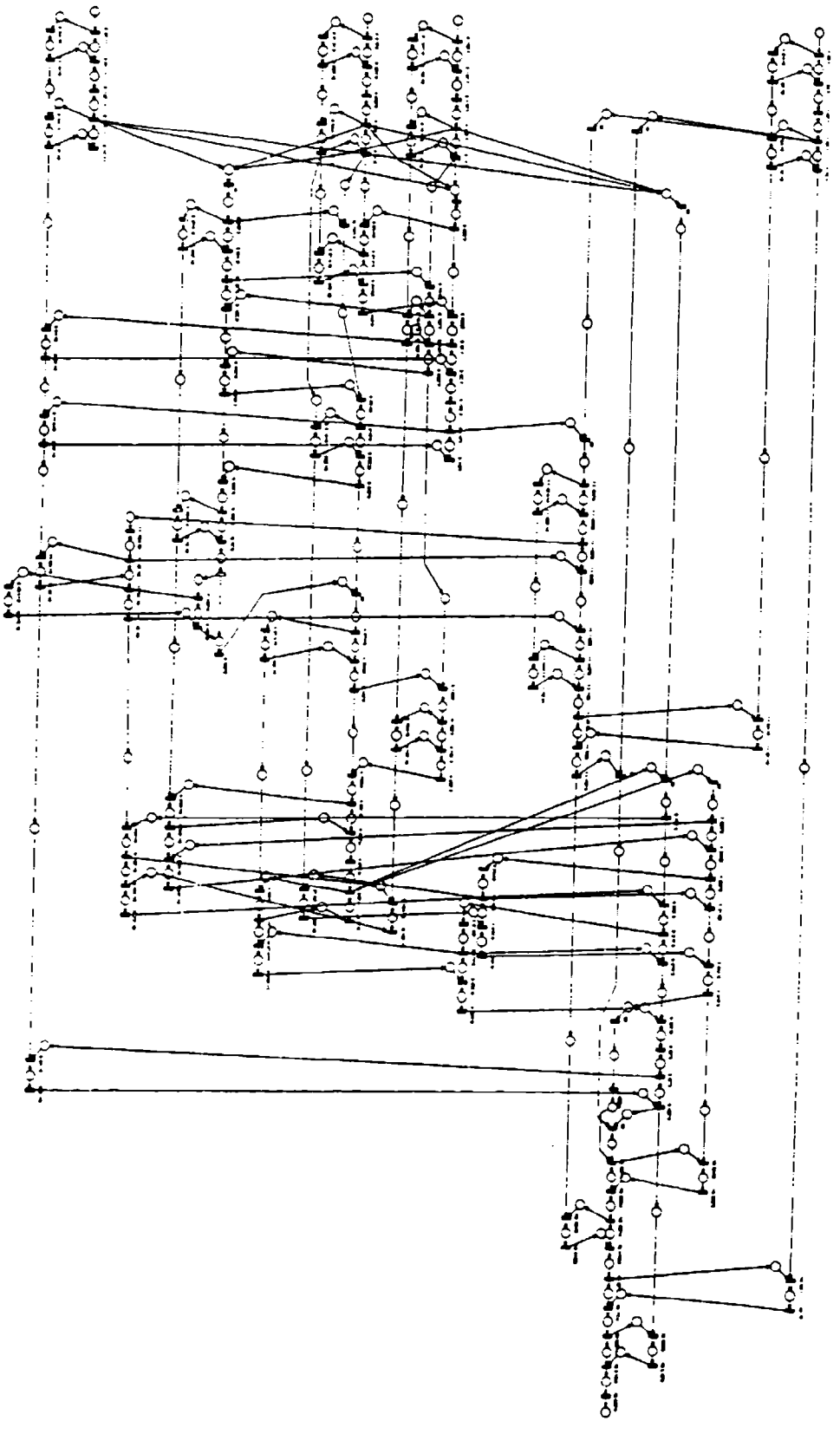


Figure 5.14 Detailed Requirements for MESACC

CHAPTER 6

CONCLUSION AND DIRECTIONS FOR FUTURE RESEARCH

6.1 CONCLUSION

Cube Tool has been extended from functions to systems. A methodology for deriving structural requirements has been proposed. Cube Tool allows to define the functions and how the various tasks, both processing and communications, are allocated to the different actors of the system and how these actors interact. The Cube Tool Responsibility Analysis plane is the basis for generating the Petri Net of the responsibilities for each function. These nets are merged according to certain rules to obtain a representation of the scenario-dependent requirements which include the processes and the communications. Different modes can be taken into consideration by using Colored Petri Nets. This methodology fills a gap between the description of requirements and the quantitative models needed for the analysis and evaluation of C3I system designs.

6.2 DIRECTIONS FOR FUTURE RESEARCH

For future research several directions can be followed. The first one is to generate architectures from the Petri Net representation of the requirements. "Actor" is a generic term for a set of human and hardware/software resources located at a specific place. The architecture generation process will be to define in more detail how the activities performed by a given actor can be distributed among the different available resources. This will require the addition of some resource places on the requirements net.

Once this is done, the System Effectiveness Analysis methodology can be applied to make Cube Tool a complete tool for designing C3I systems. For each of the architectures obtained, measures of performance can be evaluated to determine the effectiveness of a system. These measures of effectiveness will be another criterion for selecting the system to be built.

Another direction is to improve the methodology of Valraud and Levis (1989) for comparing a system with its requirement with the Petri Net of the requirements obtained by using the procedures described in this report. The Petri Net representation of a system is compared to a Petri Net representation of the requirements which only describes the interrelationships among

processes. By using the detailed Petri Net of the requirements, a more accurate view of the shortfalls and overlaps can be obtained.

REFERENCES

Demaël, J. J. , 1989, "On the Generation of Variable Structure Distributed Architectures," LIDS-TH-1869, Laboratory for Information and Decision Systems, MIT, Cambridge, MA.

Genrich H.J., Lautenbach, K., 1981, "System Modeling with High Level Petri Nets," *Theoretical Computer Science, 13* , pp. 109-136.

Genrich, H. J., 1987, "Predicate Transition Nets" in *Advanced Course on Petri Nets 1986*, Lecture Notes in Computer Science, Springer Verlag, Berlin.

Hillion, H. P. and Levis A. H., 1987, "Timed Event-Graph and Performance Evaluation of Systems," *Proc. 8th European Workshop on Applications and Theory of Petri Nets*, Zaragoza, Spain.

Jensen, K., 1987, "Colored Petri Nets" in *Advanced Course on Petri Nets 1986*, Lecture Notes in Computer Science, Springer-Verlag, Berlin.

Memmi, G. and Roucairol, G., 1980, "Linear Algebra in Net Theory" in *Net Theory and Applications. Proceedings of the Advanced Course on General Net Theory of Processes and Systems*, Hamburg 1979. W. Brauer Ed., Springer-Verlag, Berlin.

Monguillet, J.-M., and Levis, A. H., 1988, "Modeling and Evaluation of Variable Structure Command and Control Organizations," *Proc. 1988 Symposium on C2 Research*, SAIC, McLean, Virginia.

Peterson, J. L., 1980, *Petri Net Theory and the Modeling of Systems* , Prentice Hall, Inc., Englewood Cliffs, NJ.

Reisig, W., 1985, *Petri Nets. An Introduction*, Springer Verlag, Berlin.

Remy P., Levis A. H., and Jin Y. Y., 1988, "On the Design of Distributed Organizational Structures," *Automatica*, Vol. 24, No.1, pp 81-86.

Tabak D. and Levis A.H., 1984, "Petri net representation of Decisions Models," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-16, N0.6, pp.865-879.

Tournes, C. 1988, "Cube Tool a C3I Specification Oriented Tool," *Proceedings of the 9th AFCEA European Symposium and Exposition*, Brussels, Oct. 1988.

Valraud F. R. H. and Levis A. H. (1989). "On the Quantitative Evaluation of Functionality in C3 Systems," *Proc. 1989 Symposium on C2 Research*, SAIC, McLean, Virginia.